НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Автоматизована система пошуку несправностей в корпоративних IT-інфраструктурах»

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-61
Бондар Олег Олегович _____

Керівник:
декан ФІОТ, д.т.н., професор
Теленик Сергій Федорович _____

Рецензент:
Заступник декана з навчально-виховної роботи,
к.т.н., доцент каф. ТК
Лісовиченко Олег Іванович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Бондарю Олегу Олеговичу

1. Тема проєкту «Автоматизована система пошуку несправностей в корпоративних IT-інфраструктурах», керівник проєкту Теленик Сергій Федорович, д.т.н, професор, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 09 червня 2020 р.

3. Вихідні дані до проєкту: автоматизована система пошуку несправностей в корпоративних IT-інфраструктурах

4. Зміст пояснювальної записки
   4.1 Провести аналіз існуючи рішень.
   4.2 Створити вимоги до об’єкта проєктування.
   4.3 Обрати та обґрунтувати теоретичний матеріал для розробки.
   4.4 Обрати та обґрунтувати програмні засоби для розробки.
   4.5 Розробити систему для пошуку несправностей.
   4.6 Описати об’єкт проєктування.
   4.7 Провести тестування системи.
   4.8 Зробити висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

5.1 Use-case діаграма.
5.2 Контекстна діаграма IDEF0.
5.3 Діаграма процесів в IDEF0.
5.4 Документована діаграма процесів в IDEF3.
5.5 Схема електрична структурна.

6. Дата видачі завдання 24 лютого 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Отримання завдання дипломного проєкту	24.02.2020-27.02/2020	Виконано
2	Пошук та вивчення літератури з питань дипломного проєкту	28.02.2020-08.03.2020	Виконано
3	Узгодження з керівником алгоритмів та методів для вирішення завдання	09.03.2020-19.03.2020	Виконано
4	Розроблення сценарію роботи системи	20.03.2020-26.03.2020	Виконано
5	Розроблення системи	27.03.2020-30.04.2020	Виконано
6	Налагодження роботи системи	01.05.2020-07.05.2020	Виконано
7	Розроблення та налагодження інтерфейсової частини програми	08.05.2020-24.05.2020	Виконано
8	Тестування готової системи	25.05.2020-07.06.2020	Виконано

Студент Олег БОНДАР

Керівник Сергій ТЕЛЕНИК

АНОТАЦІЯ

Бондар О.О. Автоматизована система пошуку несправностей в корпоративних IT-інфраструктурах. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 81 сторінку тексту, 30 рисунків, 2 таблиці, 6 формул, посилання на 16 літературних джерел, 1 додаток та 6 конструкторських документів.

Ключові слова: автоматизована система, алгоритми навчання, бібліотеки, візуалізація даних, машинне навчання, описова статистика, пошук несправностей, IT-інфраструктура.

Об’єктом розробки є автоматизована система пошуку несправностей у корпоративних IT-інфраструктурах.

Мета розробки – підвищення якості роботи корпоративних IT-інфраструктур за рахунок пошуку несправностей.

У результаті дипломному проєкту розроблено автоматизовану систему пошуку несправностей в корпоративних IT-інфраструктурах. Для побудови даної системи використані мова програмування Python, бібліотеки для аналізу даних та машинного навчання, та знання з розділу інформатики Data Science та Machine Learning.

Отримані результати роботи корисні для роботи IT-сфер. Розроблена система може використовуватись для пошуку несправностей для будь-яких елементів IT-інфраструктури.

SUMMARY

Bondar O.O. Automated troubleshooting system for corporate IT-infrastructures. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 81 pages text, 30 figures, 2 tables, 6 formulas, links to 16 literary sources, 1 annex and 6 design documents.

Keywords: automated system, data visualization, descriptive statistics, learning algorithms, libraries, machine learning, troubleshooting, IT-infrastructure.

The object of development is an automated system of troubleshooting for corporate IT-infrastructures.

The purpose of the development is to improve the quality of work of corporate IT infrastructures by searching failures.

In the diploma project was developed the system for troubleshooting for corporate IT-infrastructure. To create this system were used libraries for the Python programming language for data analysis and machine learning, and knowledge of Data Science and Machine Learning.

The results obtained will be useful for IT-spheres. The developed system can be used to find failures for different elements of the IT-infrastructure.

№ рядка	Формат	Позначення	Найменування	Кіл. листів	№ прим.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІТ61.300БАК.004 ПЗ	Пояснювальна записка	81		
6	А3	ІТ61.300БАК.004 Д1	Автоматизована система			
7			пошуку несправностей в			
8			корпоративних			
9			ІТ-інфраструктурах			
10			Use-case діаграма	1		
11	А3	ІТ61.300БАК.004 Д2	Автоматизована система			
12			пошуку несправностей в			
13			корпоративних			
14			ІТ-інфраструктурах			
15			Контекстна діаграма	1		
16	А3	ІТ61.300БАК.004 Д3	Автоматизована система			
17			пошуку несправностей в			
18			корпоративних			
19			ІТ-інфраструктурах			
20			Діаграма процесів.	1		
21	А3	ІТ61.300БАК.004 Д4	Автоматизована система			
22			пошуку несправностей в			
23			корпоративних			
24			ІТ-інфраструктурах			
25			Документована діаграма			
26			процесів.	1		
27	А3	ІТ61.300БАК.004 Д5	Автоматизована система			
28			пошуку несправностей в			
29			корпоративних			
30			ІТ-інфраструктурах			
31			Документована діаграма			
32			процесів.	1		
33	А3	ІТ61.300БАК.004 Э1	Автоматизована система			
34			пошуку несправностей в			
35			корпоративних			
36			ІТ-інфраструктурах			
37			Схема електрична			
38			структурна.	1		

					ІТ61.300БАК.004 ТП		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Бондар О.О.			Автоматизована система пошуку несправностей в корпоративних ІТ-інфраструктура Відомість дипломного проєкту	Літ.	Арк. / Аркушів
Перевір.		Теленик С.Ф.				Т	1
Реценз.						КПІ ім. Ігоря Сікорського ФІОТ гр. ІТ-61	
Н. Контр.							
Затверд.							

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматизована система пошуку несправностей в корпоративних IT-інфраструктурах»

Київ – 2020 року

ЗМІСТ

ВСТУП

IT-інфраструктура – це комплекс взаємопов'язаних систем, які забезпечують функціонування на належному рівні інформаційної взаємодії на підприємствах.

У наш час, у великих корпоративних IT-інфраструктурах формуються ідеї, які насамперед пов'язані з покращенням інформаційно-обчислювальних процесів [1]. Наприклад, це, може, бути аналіз поведінки роботи системи, що дозволяє безвідмовно надавати IT-послуги та інше. Тому сучасне управління IT-інфраструктурою спирається на такі наукові дослідження як: системний аналіз, штучний інтелект, математичні методи дослідження оптимізаційних задач, теорія інформаційних систем, сучасна теорія управління, інформаційні технології.

Для ефективної роботи компанії (підприємства) необхідно налагоджувати узгоджену роботу усіх складових інфраструктури. Для цього потрібно вирішувати ряд задач. Нижче наведено приклад основних таких задач:

– декомпозиція бізнес-процесів;
– компонування усіх даних для оцінки якості процесів;
– забезпечення надійного та ефективного функціонування IT-інфраструктури;
– забезпечення взаємодії між різними системами управління;
– зберігання даних;
– аналіз збоїв систем та їх вплив в інфраструктурі;
– прогнозування функціонування складових систем в IT-інфраструктурі;
– пошук збоїв;
– управління з ліквідацією несправностей;
– автоматизація управління IT-інфраструктурою у цілому;
– налагодження взаємозв'язку систем;

– модернізація складових ІТ-інфраструктури;

– оптимізація процесів роботи систем;

– управління змінами та конфігураціями.

Сучасна ІТ-галузь з кожним роком розвивається, системи в компаніях стають складнішими, а несправності, які виникають при роботі, дедалі важче знайти. Проаналізувавши результати роботи елементів ІТ-інфраструктури, можна знайти причини, за яких виникли помилки. А передбачивши дані, є можливість уникнути ситуації з виникненням несправностей. Саме тому обрана тема є досить актуальною.

У якості об’єкта дослідження обрано елемент ІТ-інфраструктури, у якому виникають несправності. Предметом виступає система, за допомогою, якої можна аналізувати дані про несправності елементів ІТ-інфраструктури та передбачати збої.

Вхідні дані містять таку інформацію як: температура, вологість пристрою, час його несправності, час з останніх збоїв, хто їм керував.

Метою та завданням даного проєкту є покращення рівня автоматизації в корпоративних ІТ-інфраструктурах.

Для досягнення даної цілі потрібно виконати наступні задачі:

– розглянути існуючі рішення;

– проаналізувати вихідні дані одного з елементів ІТ-інфраструктури при несправностях;

– знайти рішення для поставленого завдання.

Отримані результати можуть використовуватись при роботі з іншими системами. Це допоможе покращити роботу компанії, адже дає можливість працівникам передбачати можливі збої систем завчасно. Це збільшить час безвідмовної роботи систем.

Для рішення поставленого завдання вирішено знайти датасет пристрою, проаналізувати його, використовуючи мову програмування Python та бібліотек для аналізу даних, спрогнозувати дані за допомогою бібліотек

для машинного навчання, побудувати автоматизовану систему пошуку несправностей, яка б виконувала прогнозування даних та аналіз самостійно.

Обрані бібліотеки включають:

– sklearn – бібліотека алгоритмів машинного навчання;

– matplotlib – бібліотека для побудови графіків;

– pandas – бібліотека для обробки та аналізу даних;

– numpy – бібліотека для роботи з багатовимірними масивами;

– seaborn – бібліотека для побудови графіків;

– tk – бібліотека для побудови графічного інтерфейсу.

Серед програмних інструментів обрано:

– Pycharm – середовище для розробки на мові програмування Python;

Для аналізу даних використані міри дескриптивної (описової) статистики:

– медіана;

– середнє арифметичне;

– стандартне відхилення.

Для перевірки правдивості прогнозованих даних використовувалось обчислення похибок:

– середньоквадратична похибка;

– абсолютна похибка;

Для машинного навчання обраний алгоритм Random Forest («Випадковий ліс») та логічна регресія.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 16 найменувань, 1 додатка. Графічна частина включає 6 креслеників формату А3. Загальний обсяг 82 сторінки.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Головним завданням даної роботи є розробка автоматизованої системи пошуку несправностей в корпоративних IT-інфраструктурах.

Для реалізації автоматизованої системи пошуку несправностей (далі – АСПН) розглянуто схожі за функціоналом системи.

На сьогоднішній день існують безліч продуктових рішень. Проаналізувавши багато з них, можна виділити три найбільш популярних та якісних продуктів:

– Splunk;

– ELK;

– Graylog 2.

Нижче зроблений детальний огляд та аналіз цих систем.

1.1 Splunk-система

Splunk – система, яка займається аналізом даних у реальному часі. На рисунку 1.1 зображено фрагмент роботи Splunk.

Рисунок 1.1 – Фрагмент роботи Splunk-системи

На основі вхідних даних Splunk будує графіки, звіти-аналізу. Через API даний сервіс може напряму підключатись до пристроїв чи веб-додатків, вилучати корисну інформацію та прогнозувати дані. Даний продукт написаний на мовах програмування C/C++ та Python.

Переваги Splunk-системи наступні:

– моніторинг у реальному часі;
– аналіз даних та їх відображення;
– оповіщення про можливі несправності;
– можливість пошуку по даним;
– захищеність;
– довготривала підтримка.

Серед недоліків даного продукту є:

– обмеження на його безкоштовне використання;
– платне використання досить сильно залежить від трафіку;
– «незручний» та незрозумілий інтерфейс для користувачів-початківців.

Згідно перерахованих переваг та недоліків, даний продукт добре підходить для задач аналізу даних.

1.2 ELK stack

ELK – абревіатура трьох продуктів: Elasticsearch, Logstash, Kibana. Поєднання цих продуктів дає можливість працювати з даними, а також робити моніторинг інфраструктур [2].

Перевагами даних продуктів наступні:

– безкоштовне використання;
– відкритий код;
– легко встановлювати та конфігурувати.

Структура даної системи відображена на рисунку 1.2.

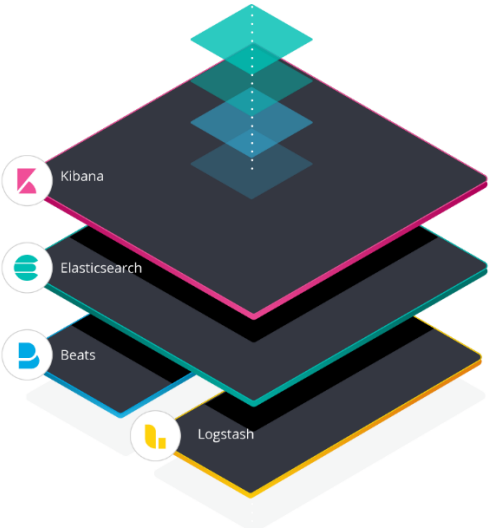


Рисунок 1.2 – Структура ELK

Детальний опис кожного з цих продуктів наведений нижче.

1.2.1 Elasticsearch

Elasticsearch – сервіс, для пошуку інформації, її збереження та аналізу. В основі лежить бібліотека Lucene. Продукт написаний на мові програмування Java, для передбачення даних використовується C/C++. На даний момент часу Elasticsearch більш стабільний та популярний у виконанні своїх задач, порівнюючи з аналогічним продуктом Solr [3]. Розберемо переваги та недоліки даного сервісу.

Переваги:

– масштабованість – можливість розширювати розмір індексування, пропускну здатність (кількість переданої інформації), кластерний розмір при збільшенні ресурсів програмного забезпечення чи апаратної частини;

– швидке виконання запитів у порівняні з багатьма аналогічними системами;

– можливість роботи з багатьма мовами програмування: Java, Python, Groovy, PHP, .NET(C#);

– документо-орієнтована система – серіалізація даних відбувається з допомогою JavaScript стандарту JSON. Elasticsearch використовує нереляційну базу даних(NoSQL).

Недоліки:

– має функцію збору та видалення даних (Garbage Collection), тобто відбувається видалення об’єктів, які довгий час не використовувались у програмі;

– не має змоги використовувати даний сервіс у якості збереження даних.

Фрагмент створення індексу для Elasticsearch наведено нижче: на рисунку 1.3.

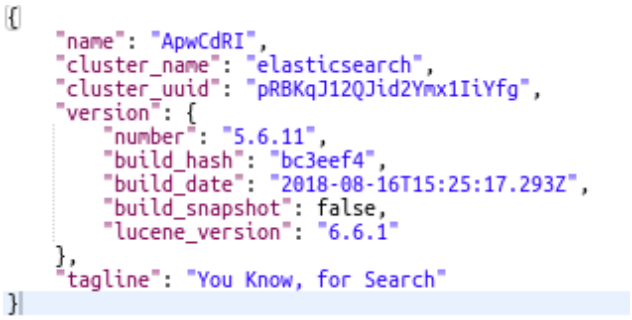


Рисунок 1.3 – Фрагмент створення індексу для Elasticsearch

З рисунку видно (див. Рисунок 1.3), що для роботи Elasticsearch потрібно створювати індекс, який використовують для пошукового двигуна. Це дає змогу швидко виконувати пошук по даним.

1.2.2 Logstash

Logstash – сервіс, який займається збором логів, їх перетворення в JSON та відправкою цих даних в Elasticsearch.

Даний сервіс займається фільтрацією даних, переведення у потрібний стан [4].

Також Logstash може збирати логи з різних ресурсів (інтернет, зв'язок, звичайні файли, бази даних) одночасно, використовуючи різні потоки.

Цей сервіс написаний на мові програмування Java та Ruby. У деяких випадках Logstash не використовують, а дані відправляють одразу ж до Elasticsearch, однак, це може призвести до можливих збоїв, оскільки логи відправляються без попередньої перевірки та фільтрації, тому можуть містити помилки. Приклад файлу конфігурації для Logstash наведено на рисунку 1.4.

```
output {
  stdout {
    type => "custom_log"
    message => "IP - %{clientip}. Full message: %{@message}. End of line."
  }
}
```

Рисунок 1.4 – Файл конфігурації Logstash

Серед переваг даного сервісу можна виділити наступні:

– захищеність – у випадку помилки при відправленні усі «нормальні» дані доставляються до Elasticsearch, а інші – чекають повторної обробки;

– модифікація даних;

– можливість відправляти різний формат даних.

Також є додатковий продукт, який пов'язаний з Logstash. Він має назву Beats – агент, який представлений у вигляді програм для збору інформацію з системних журналів та файлів. Зазвичай даний сервіс ставиться на стороні клієнта для збирання усіх логів систем. Beats є необов'язковим сервісом у ELK.

1.2.3 Kibana

Kibana – інтерфейс у браузері, що займається пошуком даних та їх візуалізацією, пов'язаний з сервісом Elasticsearch. Продукт Kibana можна використовувати у вигляді моніторингу в інших системах. Відображення

результатів роботи Kibana може бути представлений у будь-якому вигляді, починаючи з лінійних графіків, таблицями, закінчуючи гістограмами.

Фрагмент роботи Kibana наведено на рисунку 1.5.

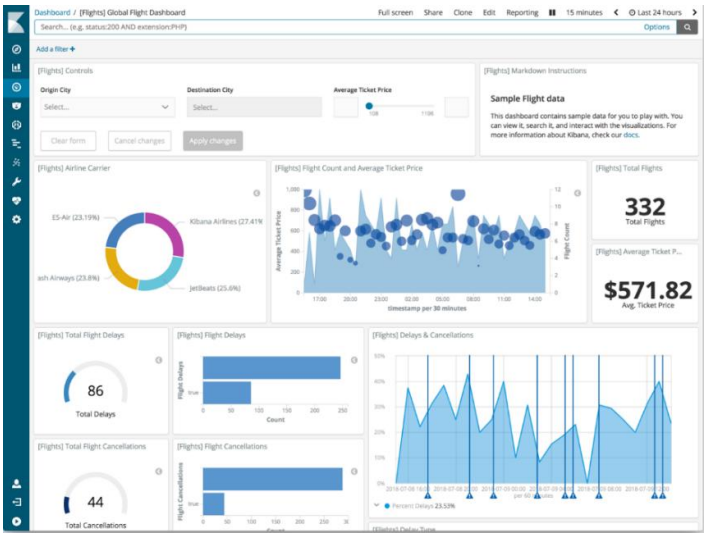


Рисунок 1.5 – Фрагмент роботи Kibana

Kibana має багато переваг:

– зрозумілий інтерфейс;

– легко змінювати візуалізацію даних;

– простіше аналізувати складні дані;

– доступно багато плагінів для полегшення роботи чи у якості додаткового матеріалу.

Головним недоліком Kibana є те, що він потребує велику кількість обчислювальних ресурсів. Однак, на сьогоднішній день, беручи до уваги розвиток обчислювальної техніки, можна вважати, що це не є досить великою проблемою, оскільки комп’ютерні можливості досить великі.

1.3 Graylog 2

Graylog 2 написаний на мові програмування Java. Використовує Elasticsearch, Logstash. Ця система призначена для кінцевого користувача. Фрагмент роботи такої системи можна переглянути на рисунку 1.6.

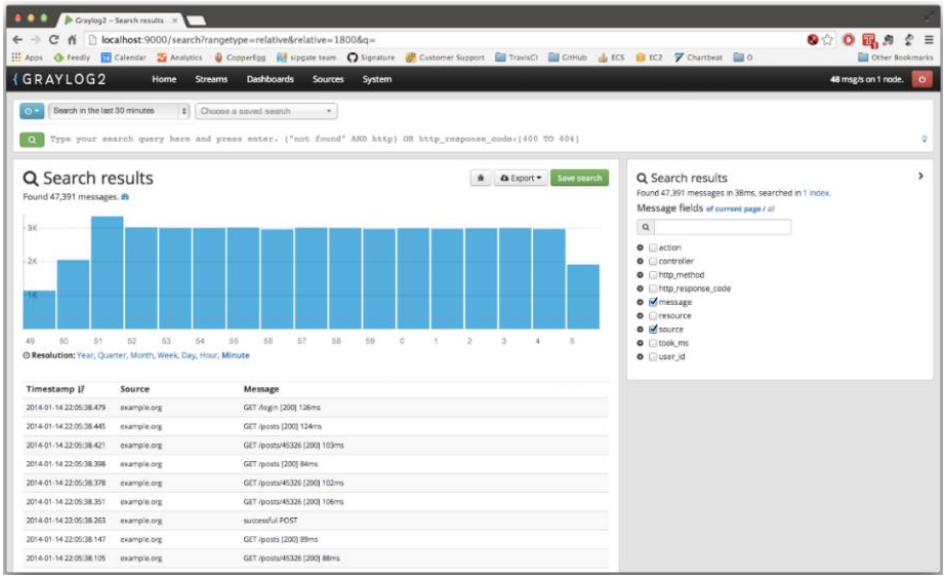


Рисунок 1.6 – Фрагмент роботи GrayLog 2

Досить схожа на ELK stack, однак відрізняться тим, що головною її метою є все ж таки збір логів. ЇЇ важко налаштувати під свої потреби, важко змінювати компоненти чи масштабувати.

1.4 Визначення вимог до реалізації системи пошуку несправностей в корпоративних IT-інфраструктурах

Розглянуті продуктові рішення мають свої переваги та недоліки, однак головною ціллю цих рішень не є пошук несправностей у системах.

Виходячи з цього, вирішено побудувати автоматизовану систему, яка б могла аналізувати дані та шукати несправності в елементах корпоративних IT-інфраструктур, беручи за основу Splunk та ELK (Elasticsearch, Logstash, Kibana). Graylog 2 базується частково на ELK, тому обирати його у якості прикладу, немає сенсу.

На основі цього рішення сформовані наступні вимоги до розроблюваної системи:

– вимоги до вхідних/вихідних даних системи;
– вимоги до аналізу даних;
– вимоги до пошуку несправностей.

Визначеним у розділі 1.4 вимогам до побудови автоматизованої системи пошуку несправностей в корпоративних IT-інфраструктурах задовольняє нижче система.

1.4.1 Вимоги до вхідних та вихідних даних системи

Нижче наведені вимоги до вхідних та вихідних даних системи:

– вхідні дані мають зчитуватись з файлу, який має розширення .csv;

– результати роботи системи (прогнозовані дані) мають записуватись в аналогічний файл з розширенням .csv;

– вхідні дані мають містити інформацію про температуру, вологість, тиск пристрою. Важливо, щоб була уся інформація про несправність обраного об’єкта дослідження: час несправності, час з останнього збою та інші.

Визначення вимог до вхідних та вихідних даних важливе для коректної роботи системи. Даним вимогам задовольняє розроблена система нижче.

1.4.2 Вимоги до аналізу даних у системі

Вимоги до аналізу даних у системі наступні:

– повинна міститись можливість обчислення показників з дескриптивної (описової) статистики. Наприклад, максимальне значення чи мінімальне значення, середнє арифметичне значення, стандартне відхилення.

– система повинна мати можливість візуалізувати дані. Серед візуальних об’єктів можуть бути звичайні лінійні графіки, гістограми, графіки-залежності величин, різноманітні діаграми (розмаху, стовбчасті, щільності) та інші.

Використовуючи аналіз даних, користувач може з’ясувати причину несправностей, тому даний функціонал та визначені до нього вимоги повинні міститись у системі.

1.4.3 Вимоги до прогнозування даних у системі

Виходячи з розглянутих існуючих рішень потрібно:

– навчити систему прогнозувати дані за допомого алгоритмів навчання;

– у системі повинні розраховуватись похибки передбачених даних. Серед таких похибок є: середньоквадратичне відхилення, середньоквадратична похибка, середня абсолютна похибка.

Варто зауважити, що для автоматизованого пошуку несправностей в елементах IT-інфраструктури потрібно звертатись до прогнозування даних, використовуючи машинне навчання, тому дані вимоги є актуальними. Представлена нижче система задовольняє їм.

Висновки до розділу 1

У даному розділі розглянуті продуктові рішення, що працюють з даними. Реалізація усіх продуктів відрізняється. Кожен з них написаний на різній мові програмування, має свою структуру, свої переваги та недоліки. Усі розглянуті продуктові рішення використовують машинне навчання для прогнозування даних, чи виявлення аномалій у даних, при цьому беручи різні технології та алгоритми.

Однак розглянуті рішення не мають функціоналу, який потрібен для пошуку несправностей, тому вирішено побудувати іншу систему, беручи до уваги розглянуті продукти. Виходячи з цього, визначено вимоги до реалізації системи пошуку несправностей.

2 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ПОШУКУ НЕСПРАВНОСТЕЙ В КОРПОРАТИВНИХ IT-ІНФРАСТРУКТУРАХ

На основі проаналізованих та оглянутих рішень, а також визначених вимог потрібно:

– розробити структурне представлення системи;
– розробити сценарій роботи системи;
– визначити механізм обробки запитів користувача.

2.1 Розробка сценарію роботи системи

Для розробки сценарію роботи системи використано програмне забезпечення StarUML та побудовано use-case діаграму, яку можна побачити на кресленику ІТ61.300БАК.004 Д1.

Як видно з діаграми, робота системи повинна починатись з вибору файлу, у якому знаходиться датасет, оскільки на основі цього здійснюється аналіз та передбачення даних. Вибір цього файлу потрібно здійснювати з допомогою графічного інтерфейсу, який передбачений у системі.

Для аналізу даних є можливість побудувати різноманітні графіки (діаграми), вказавши потрібні параметри у системі, а також розрахувати міри описової статистики.

Прогнозування даних можливе і без попереднього аналізу даних, проте найкраще зробити аналіз для зменшення кола пошуку можливих причин несправностей. Для машинного навчання у системі використовуються два алгоритми Random Forest («Випадковий ліс»), та логічна регресія.

Зауважимо, що перший алгоритм варто використовувати у випадку складних даних, другий – навпаки. Опис даних алгоритмів наведено у даній записці нижче.

2.2 Визначення механізму обробки запитів користувача у системі

Щоб уявляти роботи системи, потрібно визначити механізм обробки її запитів. Для цього побудовано діаграми, використовуючи методології IDEF0 та IDEF3.

Перша методологія слугує для описання та формалізації бізнес процесів. Визначення механізму обробки запитів починається з побудови контекстної діаграми за допомогою нотації IDEF0. Це зображено на кресленику ІТ61.300БАК.004 Д2.

За діаграмою можна побачити, що ліва стрілка, яка входить до блоку «Автоматизована система пошуку несправностей» вказує на вхідні дані. Стрілка у верхню частину блоку вказує, що управління даною системою здійснюється за допомогою алгоритмів (методів) машинного навчання та методів аналізу даних. У нижню частину входить стрілка, що вказує на механізм (мова програмування, бібліотеки), за допомогою якого, працює система. Стрілка справа означає вихідні дані з системи (графіки, прогнозовані дані, обчислені статистичні міри).

Декомпозиція контекстної діаграми, побудована також з використанням методологію IDEF0 (зображено на кресленику ІТ61.300БАК.004 Д3). Зробивши декомпозицію контекстної діаграми, видно, що дана система має можливість аналізувати вхідні дані або прогнозувати їх.

Після виконання блоку «Аналіз вхідних даних», результати йдуть до блоку «Показати результати».

Обравши прогнозування даних, виконується блок «Прогнозування даних», після чого результати йдуть знову ж таки до блоку «Показати результати».

Декомпозиція блоку «Аналіз даних» зроблена за допомогою методології IDEF3, яку зображено на кресленику ІТ61.300БАК.004 Д4.

Як можна побачити, даний блок містить два основних блоки:

– Обчислення мір статистики;

– Побудова діаграм (графіків).

Перший блок виконується раніше, тобто виводиться результати розрахованих мір, а потім є можливість будувати рафіки.

Побудова графіків відбувається у асинхронному режимі – будується один графік, далі інший, тобто не одночасно.

Декомпозиція блоку «Прогнозування даних» показана на кресленику ІТ61.300БАК.004 Д5. Механізм даного блоку наступний. Спочатку працює блок «Вибір алгоритму для навчання», далі – або блок «Вибір алгоритму Random Forest» чи блок «Вибір логічної регресії», дивлячись який алгоритм обирає користувач. Далі результати прогнозування записуються до файлу та повертаються прогнозовані дані.

Побудова моделі здійснювалась з допомогою середовища AllFusion Process Modeler.

2.3 Розробка структурного представлення системи

Для розуміння взаємозв’язку розроблюваної системи з іншими елементами ІТ-інфраструктури, а також огляд структури самої системи, побудовано структурну схему, яка зображена на кресленику ІТ61.300БАК.004 Э1.

Дана схема виконана за допомогою програмного забезпечення Visio.

Структурна схема містить наступні компоненти:

– Користувач;

– Графічний інтерфейс;

– АСПН;

– Робоча станція;

– Пристрої.

Нижче наведено докладний опис перелічених компонент, а також описано роботу автоматизованої системи пошуку несправностей.

2.3.1 Компонент «Користувач»

Компонент «Користувач» слугує для відображення взаємодії користувача (аналітика) з АСПН (автоматизована система пошуку несправностей) через компоненту «Графічний інтерфейс».

На рисунку 2.1 зображено його вигляд.

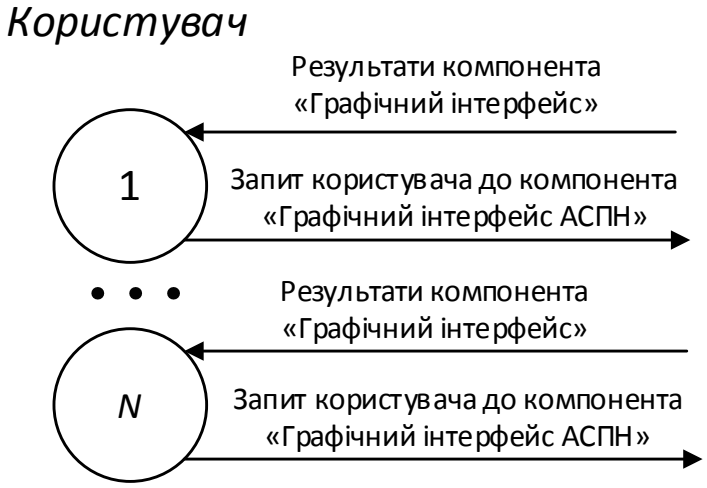


Рисунок 2.1 – Компонент «Користувач»

Користувач постійно взаємодіє з системою: повинен вводити дані для розрахунку статистичних мір, побудови графіків (діаграм, гістограм) чи для прогнозування даних.

На виході компонента «Користувач» – запит користувача до компонента «Графічний інтерфейс», на вході – «Результати компонента «Графічний інтерфейс».

2.3.2 Компонент «Графічний інтерфейс»

Даний компонент представляє собою модуль, за допомогою якого, користувач взаємодіє з системою.

Вхідними даними компонента є запити користувача, а результатами роботи є відображення графічних вікон на екрані.

Даний компонент відображений на рисунку 2.2.

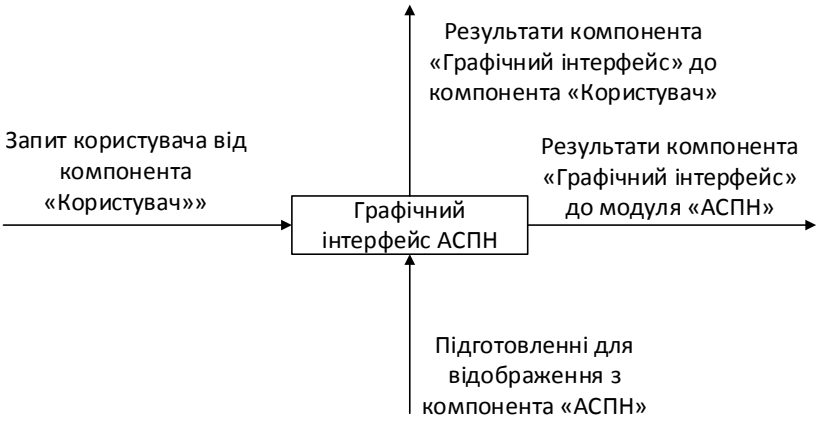


Рисунок 2.2 – Компонент «Графічний інтерфейс»

Перелік всіх вікон наведений нижче:

– стартове вікно;
– вікно для вибору функції роботи системи;
– вікно для роботи з функцією прогнозування даних;
– додаткове вікно для роботи з функцією прогнозування даних;
– результуюче вікно для функції прогнозування даних;
– вікно для аналізу даних
– результуюче вікно для функції аналізу даних.

У стартовому вікні користувач обирає потрібний файл для подальшої роботи системи.

Після стартового вікна з’являється вікно для вибору функцій роботи системи: аналіз даних чи прогнозування даних.

Обравши функцію аналізу даних, відкривається відповідне вікно для аналізу даних, при іншому виборі – вікно для роботи з функцією прогнозування.

Вікно для аналізу даних містить таблицю з обчисленими статистичними мірами, також є поля для введення параметрів, які використовуються при побудові графіків.

Спочатку усі поля у вікні недоступні для користувача, а стають доступними лише при виборі відповідних графіків. Так для побудови діаграми-помилок потрібно вводити лише один параметр, для графіків залежності, гістограми, діаграми-щільності та діаграми-розмаху – два параметри, а для побудови зведених даних – три параметри.

Після вказання потрібних параметрів, з'являється вікно з побудованим графіком (діаграмою, гістограмою).

Обравши функцію прогнозування даних, відкривається відповідне вікно, яке містить наступні поля для введення даних:

– поле для введення кількості характеристик досліджуваного об'єкта, для яких потрібне прогнозування;

– тип алгоритму для навчання.

– назва колонки, що містить інформацію про несправність.

Зазвичай значення такої колонки 0 або 1 (no або yes), тобто є вказівка про те, виникав в певний момент часу збій чи ні;

Додаткове вікно для роботи з функцією прогнозування даних містить поля, у яких потрібно ввести назви потрібних характеристик досліджуваного об'єкту, для яких потрібне передбачення даних. Кількість полів залежить від вказання цього параметру у першочерговому вікні для прогнозування даних.

Результуючі вікна для функцій аналізу даних та прогнозування даних показують вихідні дані системи: розраховані похибки, прогнозовані дані, відсоток правдивості передбачених даних, графіки (діаграми, гістограми).

2.3.3 Компонент «АСПН»

Наступний компонент «АСПН» слугує для аналізування даних та прогнозування даних на основі вхідних даних. Зображення даного компонента показано на рисунку 2.3.

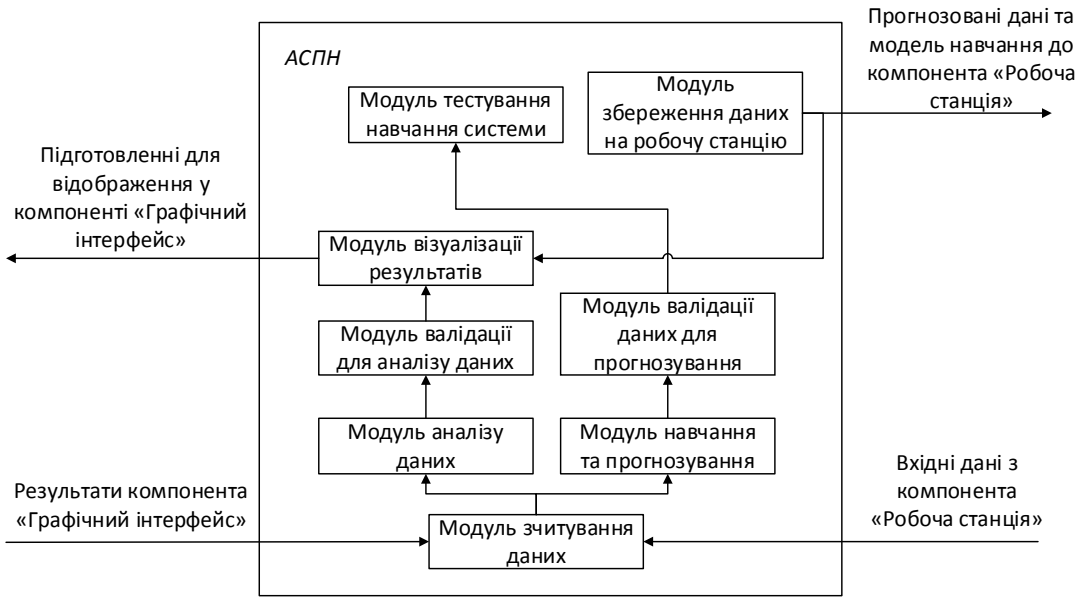


Рисунок 2.3 – Компонент «АСПН»

Як видно з рисунку (див. Рисунок 2.3) «АСПН» складається з наступних модулів:

– Модуль зчитування даних;
– Модуль аналізу даних;
– Модуль прогнозування та навчання даних;
– Модуль валідації для аналізу даних;
– Модуль валідації даних для прогнозування;
– Модуль візуалізації результатів;
– Модуль тестування навчання системи;
– Модуль збереження даних на робочу станцію.

«Модуль зчитування даних» слугує для зчитування обраних користувачем даних у форматі .csv, заповнення пустих комірок, заміна строкового представлення даних у числовий, де це є можливим.

«Модуль прогнозування та навчання даних» повинен використовуватись для навчання системи та прогнозування даних на основі вхідних даних. В основі цього модулю лежить машинне навчання.

Види алгоритмів, які використовуються у цьому модулі, описані нижче.

«Модуль аналізу даних» використовується для:

– обчислення статистичних мір описової статистики;

– побудови графіків (діаграм, гістограм).

У ньому відбувається формування графіків (діаграм, гістограм), обчислення мір статистики для подальшого відображення цих результатів у графічному інтерфейсі.

«Модуль валідації для аналізу даних» та «Модуль валідації даних для прогнозування» використовується для перевірки коректності введених даних.

Нижче наведений перелік усіх перевірок у системі:

– на невід’ємне число менше шести;

– на пусте значення параметру;

– на коректність параметрів зі строковим представленням;

Основна ціль «Модуля тестування навчання системи» полягає в том, що у ньому обчислюються похибки, а також відсоток правдивості прогнозованих даних.

«Модуль збереження даних на робочу станцію» слугує відповідно до збереження результатів прогнозування. Серед результатів збереження є модель навчання системи, а також передбачені значення. Усі дані зберігаються у компоненті «Робоча станція» у «Сховищі для прогнозованих даних», у кореневій папці операційної системи (Unix-подібних чи Windows).

«Модуль візуалізації результатів» слугує для відображення результатів роботи системи. Даний модуль працює, виходячи з того, який функціонал обрав користувач. Тобто, обираючи прогнозування даних, під час роботи «Модуля збереження даних на робочу станцію» дані передаються у «Модуль візуалізації результатів». У випадку вибору функції аналізу даних «Модуль валідації для аналізу даних» передає результати «Модулю візуалізації результатів».

2.3.4 Компонент «Робоча станція»

Даний компонент складається з наступних складових:

– Сховище для прогнозованих даних;

– Сховище даних, яке містить інформацію про збої.

На рисунку 2.4 зображено компонент «Робоча станція».

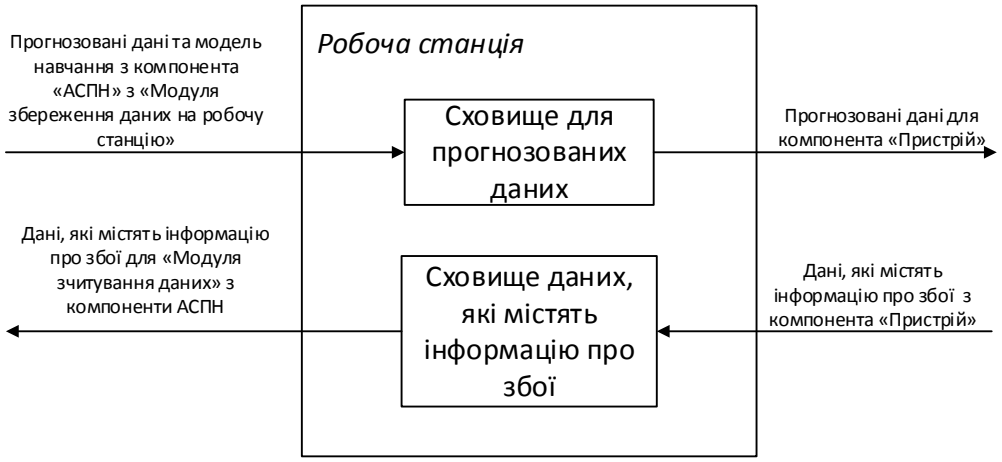


Рисунок 2.4 – Компонент «Робоча станція» на структурній схемі

«Робоча станція» може бути представлена у вигляді сервера чи персонального комп’ютера з будь-якою операційною системою: Windows чи Unix-подобною системою.

У «Сховище для прогнозування даних» записуються результати під час виконання «Модуля збереження даних на робочу станцію».

У «Сховище даних, які містить інформацію про збої» записуються усю інформацію про несправності окремих пристроїв.

Обидва сховища представлять собою деяку директорію чи папку.

2.3.5 Компонент «Пристрій»

Компонент «Пристрій» може бути у вигляді будь-якого пристрою.

Серед пристроїв можна виділити наступні:

– датчики;

– станки;

– регулятори;

– ґаджети;

– комп’ютери;

– контролери.

Усі несправності, які виникають у приладі записуються у журнал логів та відправляється у компонент «Робоча станція» у «Сховище даних, що містить інформації про збої». На рисунку 2.5 зображено даний компонент.

Компонент «Пристрій» має дві складові:

– Збереження логів;

– Зчитування даних для використання.

Рисунок 2.5 – Компонент «Пристрій» на структурній схемі

Складова «Збереження логів» потрібна для збереження інформації про несправності відповідного пристрою.

Складова «Зчитування даних для використання» потрібна для зчитування результатів отриманих після роботи «Модуля для збереження даних на робочу станцію» з компоненти «АСПН».

2.3.6 Робота автоматизованої системою пошуку несправностей

Згідно структурної схеми дана система працює наступним чином:

a) збереження даних про несправності та відправка до робочої станції;

b) модуль зчитування даних;

c) модуль прогнозування та навчання даних:

1) модуль валідації даних для прогнозування;

2) модуль тестування навчання системи;

3) модуль збереження даних на робочу станцію;

4) модуль візуалізації результатів;

5) відображення результатів на графічному інтерфейсі;

6) зчитування пристроєм прогнозованих даних з робочої станції;

d) модуль аналізу даних:

1) модуль валідації для аналізу даних;

2) модуль візуалізації даних;

3) відображення результатів на графічному інтерфейсі.

Більш детальний опис роботи системи наведено нижче.

Уся зібрана інформація про несправності на пристроях за певний час збирається на приладах та відправляється у компонент «Робоча станція» у «Сховище для збереження логів».

Через «Графічний інтерфейс» користувач (аналітик) завантажує файл, що містить інформацію про збої пристрою з компоненту «Робоча станція» за допомогою «Модуля зчитування даних» та починається робота системи. Зчитування можливе лише файлів з розширення .csv, оскільки їх легше обробляти та аналізувати.

Після цього є вибір у користувача, чи використовувати функцію для аналізу даних чи прогнозування даних. Відповідно користувач може обрати одну з них.

Обравши першу функцію, починається робота «Модуля аналізу даних», з допомогою якого, обчислюється статистичні міри: медіана, середнє арифметичне, стандартне відхилення. Використовуючи даний модуль можна виводити графіки та діаграм (діаграм-залежностей, діаграм-розмаху та інші) чи гістограм.

«Модуль аналізу даних» використовує «Модуль валідації для аналізу даних», де відбувається перевірка введених даних користувачем.

У випадку правильності даних викликається «Модуль візуалізації даних», де користувач бачить результат роботи аналізу даних (відображення

графіків), у іншому випадку – відображається повідомлення про помилкові дані.

Використавши функцію прогнозування даних, починається робота «Модуля прогнозування даних», де відбувається заповнення невистачаючих даних значенням медіани.

Навчання системи відбувається відповідно до обраного користувачем алгоритму (Random Forest чи логічна регресія).

«Модуль валідації даних для прогнозування» використовується після введення користувачем даних. У разу успішності йде перехід до наступного модулю, у іншому – виведення помилки.

При успішній перевірці запускається «Модуль тестування навчання системи». У цьому модулі йде розрахунок відсотків правдивості прогнозованих даних, а також похибок: середньої абсолютної похибки, середньої квадратичної похибки та інші.

Далі йде робота «Модуля збереження прогнозованих даних, а також моделі навченої системи до компоненту «Робоча станція» у складову під назвою «Сховище для прогнозованих даних», а також відбувається робота «Модулю візуалізації результатів» з компоненти «АСПН», що формує результати роботи функції та відправляє до компонента «Графічний інтерфейс» для відображення.

У свою чергу, зі «Сховища для прогнозованих даних» дані зчитуються відповідним пристроєм з допомогою складової «Зчитування даних для використання».

Висновки до розділу 2

У даному розділі були розроблені use-case діаграма, структурне представлення системи, а також механізм обробки запитів користувача.

3 ВИБІР І ОБҐРУНТУВАННЯ МЕТОДІВ ДЛЯ ПОБУДОВИ СИСТЕМИ

На основі визначених вимог до об’єкта проєктування, а також побудованих схем, обрано теоретичні методи, а також програмні інструменти для побудови системи..

Для цього розглядались наступні питання:

– міри дескриптивної статистики;
– алгоритми для машинного навчання;
– похибки;
– мова програмування Python;
– бібліотеки для машинного навчання;
– бібліотеки для візуалізації даних та аналізу;
– бібліотеки для інженерних розрахунків;
– програмне забезпечення для роботи с мовою програмування Python;
– набір компонентів для створення десктопного графічного вигляду розроблюваної системи;
– система контролю версії.

Нижче наведений опис розглянутих питань.

3.1 Міри описової статистики

Описова (дескриптивна) статистика займається систематизацією даних, їх представленням. У інформатиці є невід’ємною частиною науки про дані або як її часто називають Data Science, що охоплює великий спектр задач, та яка пов’язана зазвичай з іншими спорідненими науками: машинним навчанням (Machine Learning), наукою про мислення (Cognitive Science), наукою про великі дані (Big Data).

У даному пункті розглянуті основні міри дескриптивної статистики, які найчастіше використовують при аналізі даних:

– середнє значення;

– стандартне відхилення;

– медіана;

– мода.

Наступні підпункти містять опис цих мір: формули, їх призначення, переваги та недоліки.

3.1.1 Середнє значення

Середнє (просте чи арифметичне) значення вибірки, яке обчислюється за формулою (3.1) – розрахунок середнього арифметичного.

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}, \tag{3.1}$$

де x_n – елемент у дискретному ряд;

n – кількість елементів ряду.

Перед тим, як обрати дану міру для аналізу даних, потрібно оглянути переваги та недоліки, а також, у яких випадках варто її використовувати.

Серед переваг можна виділити наступні:

– легко розраховувати та просто зрозуміти;

– враховує усі значення розподілу.

Недоліками даної міри є:

– можуть видавати абсурдні значення. Наприклад, 1.5 телефона чи 2.5 людини. Тому варто уважно використовувати;

– на значення цього параметру впливають екстремальні значення.

Дана міра центральної тенденції пов’язана з іншою мірою, що має назву стандартне відхилення.

3.1.2 Стандартне відхилення

Міра центральної тенденції в описовій статистиці, яка відповідає за різницю між середнім значенням вибірки та іншими значеннями цієї ж вибірки. Вважається, що якщо значення цього параметру знаходиться ближче до нуля, то розсіяння даних невеликий і середнє значення досить добре описую всю вибірку [5]. Інша назва цього параметру – середньоквадратичне відхилення. Для обчислення його можна скористатись формулою 3.2 – розрахунок стандартного відхилення.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M)^2}{n}}, \tag{3.2}$$

де n – кількість елементів вибірки;

i – індекс обраного елемента;

M – середнє арифметичне значення;

x_i – значення елементу вибірки.

Головними перевагами стандартного відхилення є:

– уникнення компенсації додатних та від’ємних значень;

– одиниці вимірювання результату та значення ознак співпадають.

3.1.3 Медіана

Медіана – значення вибірки, що знаходиться у середині вибірки. Дану міру часто називають серединним значенням.

Вона обчислюється по-різному, у залежності від кількості елементів у ряду.

У дискретному ряду медіана для непарної кількості елементів обчислюється за формулою 3.3 – розрахунок медіани для непарної к-сті елементів.

$$M_e = \frac{n+1}{2}, \quad (3.3)$$

де n – число ознак у сукупності.

Для парної кількості елементів медіана дорівнює середньому значенню з двох взятих у середині рядка.

Серед переваг у підрахунку медіани є наступні:

– легкість розрахунку цього параметру;

– для знаходження не потрібні усі значення розподілу.

Серед недоліків:

– потрібно відсортовані значення (у зростаючому чи спадному порядку);

– неможливо роботи алгебраїчні перетворення як із середнім значенням.

3.1.4 Мода

Мода – міра центральної тенденції, що дорівнює значенню найбільш повторюваному у розподілі.

– на моду не впливають екстремальні значення;

– легко визначити цю міру арифметичним способом та графічним;

– неможливе алгебраїчне перетворення;

– досить важко обчислити у випадку багатомодального чи бімодального розподілу.

3.1.5 Обґрунтування вибору статистичних мір

Для аналізу даних обрані наступні міри описової статистики:

– медіана;

– стандартне відхилення.

Середнє арифметичне значення та мода не завжди дають точний результат, тому ці міри не варто використовувати у питаннях аналізу даних.

Також існують такі міри: максимальне та мінімальне значення. Однак, вони також не дають точний аналіз, оскільки є поняттями відносними.

3.2 Алгоритми машинного навчання

Для знаходження та уникнення несправностей у досліджуваному об’єкті потрібно використовувати такий розділ як Machine Learning (ML).

ML – це галузь штучного інтелекту в галузі інформатики [6]. Машинне навчання будується на основі математичної моделі, яка базується на так названих «даних для навчання» для того, щоб робити припущення та передбачення майбутнього стану досліджуваної системи.

ML ділиться на дві категорії: «навчання з вчителем» та «навчання без учителя». Дані алгоритми мають відмінності та підходять для різних типів задач. Розберемо кожний з категорій окремо.

3.2.1 Навчання без учителя

Даний клас алгоритмів призначений для даних, які не мають чіткого виділення ознак, тобто модель не знає, що робити з цими даними. У цьому випадку відбувається пошук кореляції між даними, виділення корисної інформації [7]. Характерними задачами для навчання є наступні:

– асоціація – тип задач, які встановлюють зв’язок між різними даними. Наприклад, купуючи товари для прибирання: пилосос, ганчірку через інтернет-магазин, веб-додаток може запропонувати швабру для прибирання, оскільки усі товари відносяться до однієї категорії.

– кластеризація – даний тип задач у своїй основі містять пошук схожих ознак у даних. Наприклад, знаходження спільних ознак в описаних

машинах. Групування може бути за кольором, маркою, країною-виробником, за типом машини, за маркою машини та інші.

– пошук аномалій – це знаходження таких даних, що дуже сильно відрізняються від інших. Наприклад, у даних, що описують роботу певного пристрою зазначається «дивна» різниця між температурою за дуже короткий проміжок часу. Причиною може бути чи неполадка вимірювального пристрою, чи сам прилад – несправний.

У навчанні без учителя є свої переваги та недоліки. Зокрема, важко обчислити точність знайдених (передбачених даних), однак отримані дані зазвичай надійні, без зайвих шумів.

3.2.2 Навчання з вчителем

Зазвичай для такої категорії навчання потрібні розмічені дані, тобто у яких виділені основні ознаки, чи показані результати при повному можливому наборі значень певних характеристик. Такими даними можуть виступати фотографії, на яких виділені різниці між зображеними об’єктами, чи логи результатів системи, при використані тих чи інших значень. Серед задач, характерними для такого типу навчання, можуть бути:

– регресія – тип задач, які стосуються неперервних даних, в основі яких, лежить залежність однієї величини від іншої. Наприклад, якщо обрати певні значення параметрів системи, то може виникнути збій. В даному випадку залежить результат роботи системи від обраних значень характеристик;

– класифікація – тип задач, у яких передбачення даних відбувається за рахунок їх класифікації. Наприклад, на фотографіях з зображенням якогось предмету чи істоти є помітка, у якій зазначено назву зображеного предмета чи істоти. Тоді алгоритм бере наступні зображення, порівнювати з іншими і у відповідність ставити значення мітки (класифікатора) даних;

Головним недоліком даного навчання є розмічені дані, оскільки у них трапляються зазвичай велика кількість помилок, а також такі дані досить дорого коштують. Однак, обчислити точність результуючих даних досить легко.

3.2.3 Обґрунтування вибору алгоритмів для машинного навчання

Оскільки навчання базуватись на вхідних даних, які описують роботу пристрою за певний проміжок часу, тому обрано «навчання з вчителем», бо датасет містить розмічені дані.

Серед алгоритмів для навчання обрані логічна регресія, а також Random Forest («Випадковий ліс»), оскільки система у майбутньому розвиватиметься та аналізувати не тільки прості датасети, а й складніші. Тому перший алгоритм слугуватиме для одних типів задач, другий – для інших задач.

Логічна регресія – найбільш зрозумій та популярний алгоритм. Зазвичай даний алгоритм використовується для бінарної класифікації. Тобто передбачувані дані мають два значення тільки 0 та 1 [8]. Оскільки датасет, який обробляє система, містить поле «несправність», яке має значення 0, якщо немає помилки та 1, у випадку помилки, тому цей алгоритм є досить актуальним.

Він схожий на лінійну регресію, однак у якості функції для береться нелінійна у вигляді великої літери S.

По осі Оу відображається вірогідність виникнення певної події, по Ох – результуюче значення рівняння регресії.

Random Forest («Випадковий ліс») – алгоритм, який використовується для більш складних аналізів та передбачень даних [8]. Тренувальні дані розбиваються на більш менші вибірки, для яких будуються свої моделі. Приклад розбиття показано на рисунку 3.1.

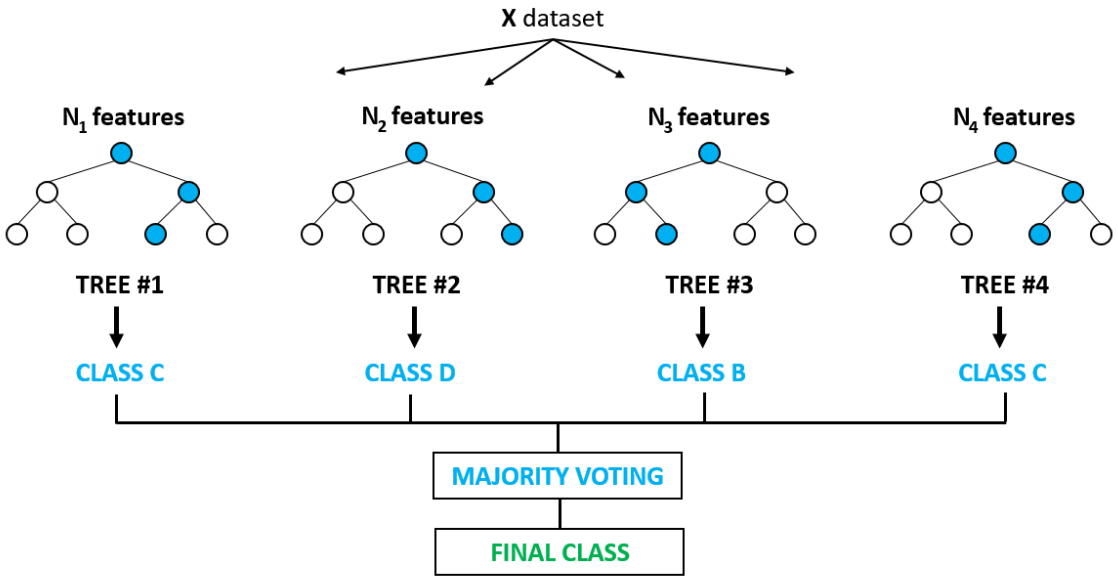


Рисунок 3.1 – Зображення алгоритму «Випадковий ліс»

Після розбиття кожна модель видає значення, для яких знаходить середнє арифметичне. Це і є результат роботи даного алгоритму.

3.3 Розрахунок похибок

Перевірка правильності передбачених даних – важлива частина для машинного навчання. Для цього використовується розрахунок похибок для результатів роботи алгоритмів.

Це потрібно для того, щоб не допустити перенавчання системи чи навпаки – недонавчання. Серед похибок виділяють наступні: середня абсолютна похибка (MAE), середня квадратична похибка (MSE), корінь із середньо-квадратичної похибки (RMSE).

3.3.1 Середня абсолютна похибка

Середня абсолютна похибка (MAE) – вид похибки, який не чутливий до викиду даних [9]. Зазвичай використовується для вимірювання неперервних даних. Обраховується за формулою 3.4 – розрахунок MAE.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}, \quad (3.4)$$

де y_i – передбачене значення;

x_i – правильні дані;

n – кількість значень у вибірці.

Для даного виду помилку, чим менший результат, тим точніші передбачувані дані.

3.3.2 Середня квадратична помилка

Середня квадратична помилка (MSE) – міра, яка найбільше використовується. Головним недоліком цього параметру є чутливість до шумів у наборі даних [8].

Шум даних – велика кількість значення, які невірно описують вибірку і тому дають невірний результат при аналізі. Наприклад, температура тіла людини 45° чи зріст 3 м.

Щоб обчислити середню квадратичну помилку, потрібно скористатись формулою 3.5 – розрахунок MSE.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}, \quad (3.5)$$

де y_i – спостережувані дані;

x_i – передбачувані дані;

n – кількість значень вибірки.

Найкраще використовувати дану міру, коли містяться занадто великі чи занадто малі значення.

3.3.3 Квадратичний корінь з середньоквадратичної похибки

Корінь квадратний з середньо-квадратичної помилки – один з видів розрахунку точності передбачуваних даних.

Корінь використовується для усереднення результуючих значень Найкориснішою дана міра є у випадку, коли є великі помилки і вони впливають на модель. Це дозволяє уникнути абсолютного значення помилки [9].

Для розрахунку існує формула 3.6 – розрахунок RMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}}, \quad (3.6)$$

де y_i – спостережувані дані;

x_i – передбачувані дані;

n – кількість значень вибірки.

Для даного виду помилки вважається, щоб чим менший її результат, тим краще.

3.4 Мова програмування Python

Python – мова програмування зі строгою динамічною типізацією даних. Підтримує різноманітні парадигми проєктування, серед яких є процедурна, об’єктно-орієнтована, функціональна, імперативна, аспектно-орієнтована [10]. Увібрала усі найкращі сторони таких мов програмування, як Java, Fortran, C/C++, Lisp.

Часто дана мова використовується для розв’язання задач, пов’язаних з машинним навчанням, для складних інженерних розрахунків та аналізу даних. Для перелічених задач існують чимало бібліотек. Серед найкращих це: pandas, matplotlib, sklearn, numpy, seaborn. Ці бібліотеки описані у наступних пунктах.

Python містить чимало переваг. За допомогою цієї мови можна створювати не тільки веб-додатки, але й десктопні застосування чи програми для вбудованих систем.

Програми, які використовують дану мову, набагато менші у порівнянні з додатками, що написані на інших мовах. Код добре читається.

Містить ця мова таку форму управління пам'яттю, як Garbage Collector (GC) – автоматичне прибирання сміття. Якщо об'єкт під час виконання програми не потрібен GC автоматично видаляє його. Разом з Python на комп'ютер завжди ставиться головна бібліотека, яка дозволяє працювати з операційною системою, базою даних, веб-додатками, аналізом даних, графічним інтерфейсом програм, машинним навчанням. Може розширюватись за допомогою модулів, що написані на C/C++. Працює на різних операційних системах (Windows, Linux, Mac, Raspberry).

Однак, найбільшим недоліком Python є повільніша, ніж у інших мовах програмування, швидкість виконання програми, бо це інтерпретована мова. Спочатку код перекладається у байт-код, потім обробляється інтерпретатором і запускається програма.

Основною відмінністю інтерпретатора від компілятора є та, що перший перекладає високорівневу мову на машинну по рядку і одразу виконує код., а другий за один раз перекладає увесь код, а потім тільки виконує його.

На рисунку 3.2 показано механізм роботи Python.

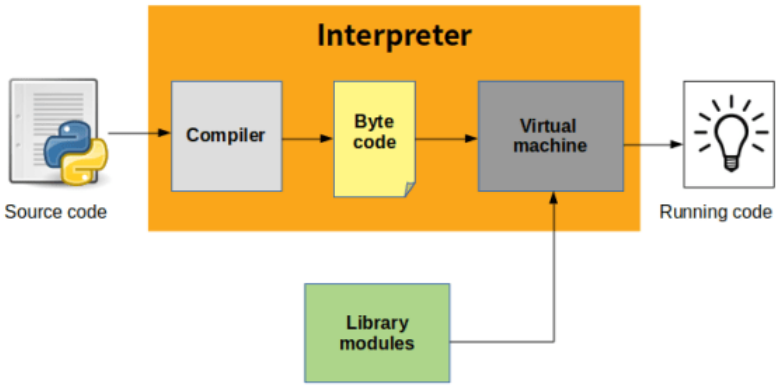


Рисунок 3.2 – Робота програми на Python

Виходячи з усіх перерахованих переваг та недоліків, можна дійти до висновку, що дана мова добре підходить для побудови АСПН, тому що є досить простою, містить багато функціоналу та написаних для неї бібліотек (для аналізу, розрахунків машинного навчання), легка у розумінні.

3.5 Бібліотеки для машинного навчання

Бібліотеки для машинного навчання – невід'ємна частина для реалізації нашої системи. На даний момент часу існує п'ять найкращих бібліотек: TensorFlow, Theano, Scikit-learn, Keras, PyTouch. Опис кожної з них наведений нижче.

3.5.1 TensorFlow

TensorFlow – безкоштовна, з відкритим кодом платформа для машинного навчання. Вона є досить гнучкою, вміщує велику кількість інструментів. Має декілька рівнів абстракції [11].

Будування та тренування моделей здійснюється за допомогою високорівневого API Keras.

Дану бібліотеку використовують разом з іншими програмами для машинного навчання.

Написана TensorFlow на Python. Доступна на будь-якій операційній системі.

3.5.2 Theano

Бібліотека і оптимізаційний компілятор для маніпулювання та оцінки математичних виразів, яка використовується для машинного навчання [10].

Вона слугує для швидких математичних обчислень, особливо для значень матричного вигляду. Може бути запущена, як на CPU, так і GPU.

Ця бібліотека також безкоштовна та з відкритим кодом. Використовує синтаксис NumPy.

3.5.3 Scikit-learn

Безкоштовна бібліотека, у якій, увесь опір робиться на алгоритми машинного навчання, зокрема:

- метод опорних векторів;
- кластеризація;
- градієнтний бустинг;
- метод К-середніх
- класифікатор;
- логічна та лінійна регресія.

У порівнянні з TensorFlow, дана бібліотека має більш простий функціонал, однак, це не заважає робити аналіз даних та її передбачення [12]. Більша частина функціоналу написана на Python, а також використовує бібліотеку NumPy для виконання складних математичних операцій, пов’язаних з лінійною алгеброю та операціями з масивами (матрицями).

Однак є деякі алгоритми, написані на Cython – програмній мові, яка заснована на Python та C, для розширення існуючих модулів.

Також такі алгоритми, як логічна регресія та лінійна регресія є обгортками над бібліотекою з відкритим кодом Liblinear.

Scikit-learn досить добре взаємодіє з іншими бібліотеками Python: matplotlib, plotly, numpy, pandas, scipy та іншими.

3.5.4 PyTorch

Проєкт з відкритим програмним кодом, основа якого Torch – бібліотека для машинного навчання, написана на мові програмування Lua.

Глибоке навчання – сукупність методів для машинного навчання (з вчителем, з частковим вибором вчителя, без вчителя, з підкріпленням), яке не зосереджене на конкретних алгоритмах.

Зазвичай PyTorch використовується для наступних додатків: обробка звичайної мови, комп'ютерний зір.

3.5.5 Keras

Бібліотека для роботи з нейромережами. Написана вона на мові програмування Python.

Призначена для роботи з мережами глибокого призначення.

Містить у собі функціонал вище описаних бібліотек: TensorFlow, Theano та CNTK. Підтримує у собі різний тип мереж: рекурентні та згорткові.

3.5.6 Вибір та обґрунтування бібліотек для машинного навчання

Усі описані бібліотеки для машинного навчання є досить корисними, однак для вирішення поставлених задач, найкраще підходить Scikit-learn, оскільки, головне призначення її, це робота з алгоритмами для навчання.

Тобто, бібліотека надає можливість обирати алгоритми, які потрібно для вирішення конкретної задачі. Тим паче Scikit містить усі потрібні алгоритми для реалізації розроблюваної системи.

3.6 Бібліотеки для візуалізації даних та аналізу

У цьому пункті розглянемо бібліотеки для аналізу даних та побудови графічного представлення.

Серед таких бібліотек на сьогоднішній день виділяють наступні:

– pandas;

– matplotlib;

– plotly;

– seaborn.

Опис даних бібліотек наведений нижче.

3.6.1 Pandas

Pandas – бібліотека для аналізу даних та візуалізації. Пов'язана з іншим пакетом NumPy. Використовують часто у немаркованих даних. Більша кількість коду написана на Python, інше частина – на Cython та C. Ця бібліотека містить такі можливості:

– для роботи з даними використовується DataFrame, структура якого на рисунку 3.3;

–

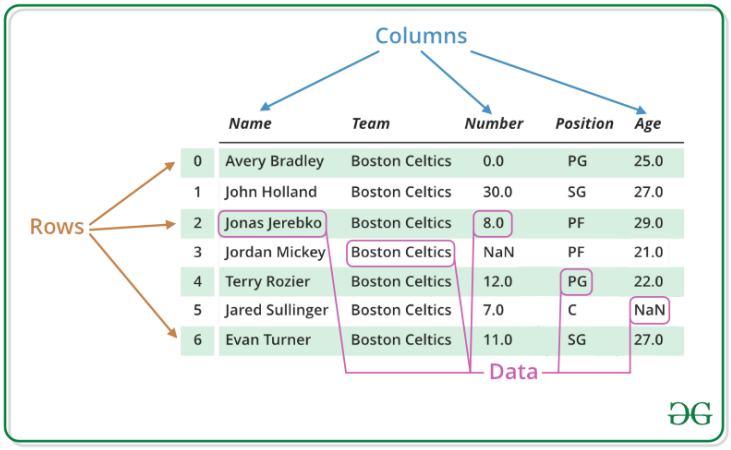


Рисунок 3.3 – Структура типу даних DataFrame

– вирівнювання даних та обробка відсутніх даних;

– зрізи даних на основі міток;

– робота (зчитування та запис) з різним форматом даних: JSON, HTML, SQL, CSV, Pickle;

– зведення усіх датасетів до одного загального датасету за аналогією, як це робиться в базах даних через JOIN команду, тому достатньо

лиш мати файл з результатами роботи пристрою, та необов’язково тримати датасет у базі даних.

– групування даних за короткий час та за найменшою кількістю рядків;

– редагувати стовбці а бо видаляти у структурі датасету;

– аналізувати ієрархічні дані;

– фільтрація даних;

– обчислення значень математичної статистики.

Зазвичай дана бібліотека працює разом з Matplotlib, яка описується нижче [13].

3.6.2 Matplotlib

Низкорівнева бібліотека для виведення графіків (діаграм, гістограм, спектрограм).

Приклади графіків зображені на рисунку 3.4 (а-в).

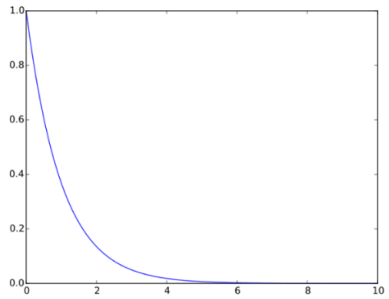


Рисунок 3.4 (а) – Приклад лінійного графіку

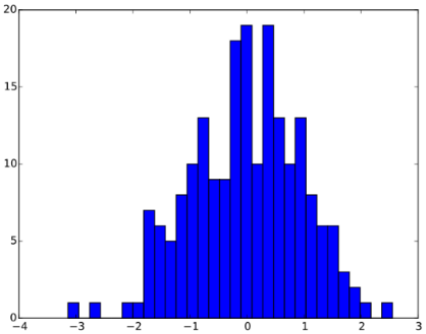


Рисунок 3.4 (б) – Приклад стовбчастої діаграми

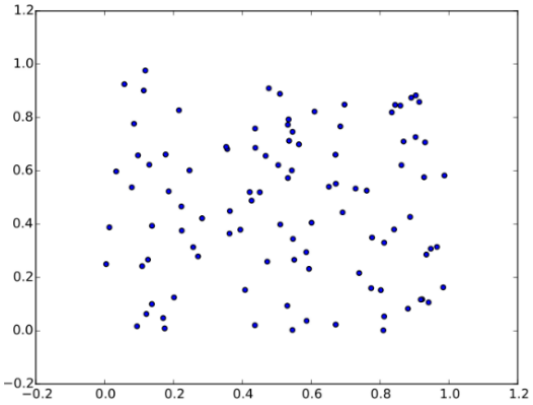


Рисунок 3.4 (в) – Приклад діаграми-розсіяння бібліотеки Matplotlib

Використовує мову програмування Python та бібліотеку NumPy. Є можливість вбудовувати у графічне відображення програмного забезпечення, наприклад, у Tkinter, wxPython, Qt чи GTK+ [14]. Займає провідне місце у візуалізації даних. Багато інших схожих бібліотек базується на ній.

3.6.3 Plotly

Бібліотека з відкритим кодом, написана на Python та фреймворку Django, яку використовують онлайн.

Графіків, які можна будувати з допомогою цієї бібліотеки зображено на рисунку 3.5.

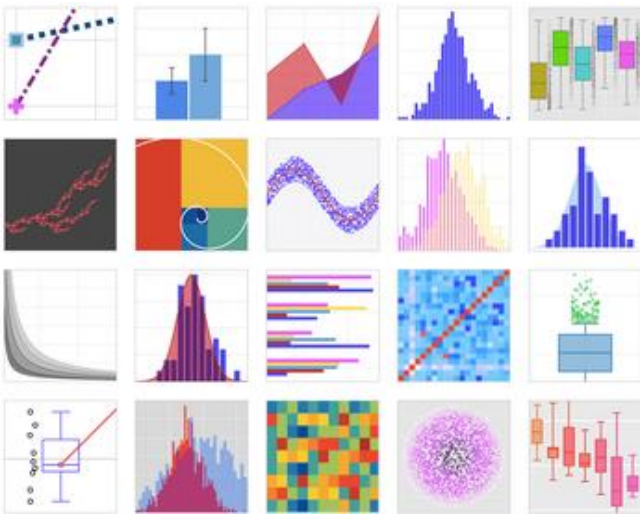


Рисунок 3.5 – Приклади побудованих графіків Plotly

Frontend частина цього інструменту написана на JavaScript, а також використовує іншу бібліотеку D3.js [15]. Головним недоліком Plotly є його виконання на стороні сервера, що робить виконання запитів досить повільно.

3.6.4 Seaborn

Бібліотека більш високо рівня, порівнюючи з matplotlib, яка дозволяє будувати карти тепла, діаграми розмаху, кластерні карти, зведені діаграми, результати математичних операцій та інші. Seaborn містить наступний функціонал:

– можлива візуалізація одномірних та двомірних розподілів та порівняння їх між підмножинами даних;

– автоматична оцінка і побудова лінійної регресії для різних залежних змінних;

– підтримується відображення результатів параметрів статистики.

Приклад діаграми-розмаху відображено на рисунку 3.6.

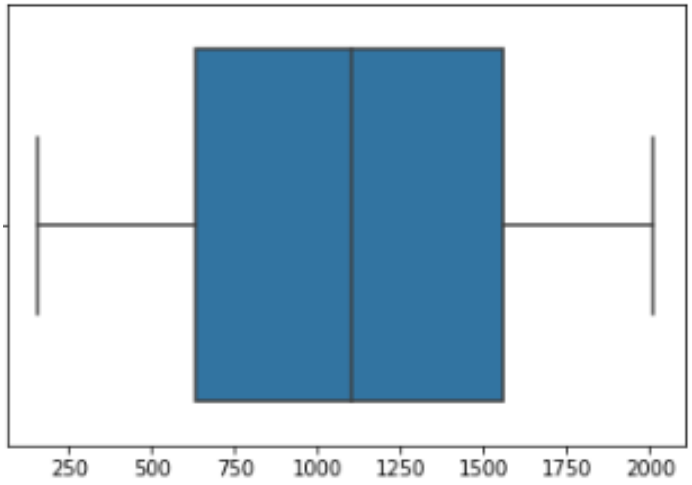


Рисунок 3.6 – Діаграма-розмаху бібліотеки Seaborn

– можливість побудови складних наборів даних є можливість вибору кольорів для відображення даних.

3.6.5 Обґрунтування вибору бібліотек для аналізу та візуалізації даних

Описані вище бібліотеки містять обширні можливості. Для аналізу даних найкраще підходить pandas, оскільки аналогів до цієї бібліотеки немає, та також вона досить оптимізована, містить потрібний функціонал та досить добре підходить для побудови АСПН в рамках цього дипломного проєкту.

Що стосується розглянутих інструментів для візуалізації даних, то краще обрати бібліотеки Seaborn та Matplotlib, бо дані бібліотеки можна використовувати не тільки для веб-застосунків як Plotly, але й для побудови десктопних додатків. Ці дві бібліотеки обрані згідно з тим, що перша містить можливість будувати специфічні графіки, діаграми для візуалізації даних, наприклад, діаграма розмахів. Інша ж дозволяє будувати інші відображення, які є більш простішими.

3.7 Бібліотека для інженерних розрахунків

Для побудови системи скористаємося бібліотекою NumPy, оскільки вона оптимізована, виконання її функції досить швидке, на ній базуються багато інших, а також вона повністю підходить для розробки АСПН. Код даної бібліотеки написаний на Python. NumPy зазвичай використовується для наукових розрахунків. Її можливості наступні:

– працює з масивами N-розміру (додавання, віднімання, множення, ділення, транспортування, обчислення визначника;

– можливість обчислювати значення параметрів лінійної алгебри та складних математичних функцій;

– перетворення Фур’є;

– генератор випадкових чисел.

Формат зберігання даних цієї бібліотеки є стандартом для зберігання числових даних в інших бібліотеках, зокрема, Pandas, Scikit-learn, SciPy та інших.

На основі описаних можливостей NumPy, можна дійсно переконатись, що вона найбільше підходить для вирішення поставлених задач, оскільки ця бібліотека взаємодіє з вже обраними, а також є можливість обчислення математичних функцій, маніпуляцій з N-мірними масивами. Останнє досить важливе аналізу даних.

3.8 Компоненти для створення графічного вигляду системи

Для створення графічного інтерфейсу існує бібліотека Tk, що містить графічні компоненти, та яка написана на мові програмування Tcl [16].

Графічний інтерфейс користувача представляється у вигляді вікон, кнопок, випадаючих списків, текстових полів, радіо кнопок чи кнопок з вибором декількох варіантів. Є можливість для початкового розміщення цих компонентів, збільшення/зменшення розмірів та інші. В основі цього лежить подійно-орієнтоване програмування, головне завдання якого, «відповісти» на викликану подію зі сторони, наприклад, користувача. Серед подій можливі наступні:

– натиснення на кнопку для підтвердження дії чи виклик вікна іншого, або відображення даних;

– подія, яка виникає у випадку написання тексту у відведеному полі;

– автоматичного оновлення, яке виникає при зміні даних (видаленні, редагуванні, додаванні)

Щоб користуватись Tk бібліотекою, потрібно використовувати Tkinter – пакет, який перетворює мову програмування Python у Tcl та дозволяє виконати програму.

Дана бібліотека є досить гнучкою, програми, написані на ній, працюють на будь-якій операційній системі, що містить встановлений Python, має дуже простий синтаксис, що дозволяє швидше та якісніше розробляти програмне забезпечення.

3.9 Програмні засоби для розробки на Python

Швидкість розробки часто залежить від програмного забезпечення, у якому, і відбувається написання програмного коду. На сьогоднішній день існують багато редакторів з вбудованими плагінами для Python або просто IDE.

Для розробки найкраще підходять два інструменты: Jupyter (Jupyter Lab та Jupyter Notebook), PyCharm.

Jupyter – це IDE, яка використовується часто для розробки на Python. Входить до складу дистрибутиву Anaconda.

Головними перевагами даної IDE є її функціонал, підтримка великої кількості мов програмування, інтерактивність. Також є можливість створювати документи, у яких одразу написаний код, рівняння, текст та результат виконання певної дії (відображення графіків чи передбачених даних та інше).

Даний інструмент підходить найкраще всього для створення звіту по аналізу даних (первинної та повноцінної обробки) та перевірки окремих написаних функцій.

Щодо розробки автоматизованої системи з графічним інтерфейсом, то найкраще підходить інший інструмент – PyCharm.

Це крос-платформова IDE, яка допомагає розробникам швидше писати софт, оскільки має можливість рефакторити код, запускати дебаг своєї програми, підсвічує синтаксис, автоматично форматує написане.

В PyCharm можна робити повноцінне тестування розробленої системи. Має велику кількість плагінів, зокрема, для роботи з контролями версії: Git, Mercurial чи з базами даних: PostgreSQL, Mongo та іншими.

Є можливість відкрити документацію для даної мови у вікні редактора, що є досить зручно, ніж шукати інформацію через браузер.

3.10 Система контролю версій Git

Щоб реалізувати АСПН, а також покращувати цю систему у майбутньому, потрібно використовувати систему контролю версії.

Для цього використовується Git. Він є безкоштовним, багатофункціональним, більш стабільним, а також зручний. Git дозволяє слідкувати за змінами у програмному коді, робити відкати коду (у випадку виникнення помилок) або інші зміни.

Для виконання основного завдання потрібно вирішувати ряд підзадач. За допомогою системи контролю версії можливо це зробити набагато легше, оскільки для кожної підзадачі можна створювати свою гілку, а результати поєднувати у головній гілці. Це дасть змогу не плутатись під час програмної реалізації якогось проєкту.

Збереження коду проєкту досить важлива частина. На сьогодні виділяють три основні сайти для хостингу програмного коду: GitLab, GitHub, Bitbucket.

Перший зазвичай використовують для комерційних цілей. Серед усіх наведених сайтів для хостингу він є найкращим, бо має багато функцій, у ньому легко працювати, а головне – зручно.

Bitbucket – безкоштовний сервіс, однак набагато простіший, ніж інші два, тобто не дає повних можливостей для розробки.

GitHub – веб-сервіс, який безкоштовний. У ньому є можливість мати як приватні репозиторії, так і репозиторії з відкритим доступом. Найкраще підходить для розробки продуктів з відкритим програмним кодом, підходить як для комерційних цілей, так і власного користування, для навчання, досить інтуїтивний.

Описані сервіси для хостингу мають свої переваги та недоліки, однак для виконання поставлених задач, найкраще підходить GitHub, бо є безкоштовним, легкий у використані, дозволяє публікувати проєкти з відкритим кодом. Серед системи контролю версії обрано Git.

Висновки до розділу 3

Даний розділ був присвячений вибору теоретичних методів, а також інструментів для побудови автоматизованої системи пошуку несправностей в корпоративних IT-інфраструктурах.

Для аналізу даних обрано розрахунок мір описової (дескриптивної) статистики: медіани, стандартного відхилення. Результати даних величин дадуть можливість зробити первинний аналіз вхідних даних.

Щоб прогнозувати дані та уникати несправності у подальшій роботі досліджуваного елементу IT-інфраструктури, використовується машинне навчання. Серед алгоритмів для даного функціоналу обрані алгоритми Random Forest («Випадковий ліс») та логічна регресія, які входять до групи «навчання з вчителем».

Після виконання алгоритмів бажано перевіряти результати, щоб не допустити можливості перенавчання системи чи навпаки її недонавчання. З цією ціллю обрано формули для обчислення середньої абсолютної помилки, середньої квадратичної помилки.

Для реалізації системи, обрано мову програмування високого рівня Python. Допоміжними для неї бібліотеками є наступні:

– Pandas;
– Matplotlib;
– Seaborn.

Обрана Scikit-learn бібліотека потрібна для вирішення задач з машинним навчанням, а NumPy – для розрахунків складних математичних функцій (виразів) та роботою з масивами.

Для побудови графічного відображення системи обрано бібліотеку Tk та відповідний пакет Tkinter для трансформування мови Python у Tcl.

IDE PyCharm обрана для програмної реалізації системи, для контролювання написаного коду – система контролю версії Git та сервіс для хостингу GitHub.

4 РЕАЛІЗАЦІЯ ОБ'ЄКТА ПРОЄКТУВАННЯ

Базуючись на теоретичних відомостях для вирішення задачі пошуку несправностей в IT-інфраструктурах та обравши інструменти для програмної реалізації системи, побудована автоматизована система пошуку несправностей.

Для опису реалізованої системи, у розділі описано функціональну її частину: користувацькі та стандартні функції, також представлене тестування цієї системи та детальний опис її покрокового використання.

4.1 Функціональна частина системи

4.1.1 Користувацькі функції

У таблиці 4.1 наведений перелік функцій, які написані для роботи користувач з системою.

Таблиця 4.1 – Користувацькі функції

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	build_error_diagram	Побудова діаграму помилок	Назва колонки, яка відповідає за помилки	Діаграма помилок
2	dictionary_sort	Сортування словника	Словник помилок	Відсортований словник

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
3	get_colors	Створення кольорів для виведення графіків	Кількість кольорів	Масив кольорів
4	read_file	Зчитування файлу	Шлях до файлу з датасетом	Дата фрейм
5	build_dependency_diagram	Побудова діаграми залежності	Назва параметру у діаграмі	Діаграма залежності помилок від вхідного параметру
6	find_statistics_param	Побудова діаграми залежності	Назва параметру у діаграмі	Діаграма залежності помилок від вхідного параметру
7	find_statistic_parameters	Пошук статистичних мір	Шлях до файлу з датасетом	Дата фрейм з обчисленими мірами
8	build_dependency_diagram	Побудова діаграми залежності	Назва параметру у діаграмі	Діаграма залежності помилок від вхідного параметру
9	replace_value_data	Заміна значень	Датасет	Змінені дані у датафреймі

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
10	data_error	Отримання даних для виводу графіків залежності	Назва будь-якої колонки	Загальна кількість кольорів, масив кольорів
11	build_pivot_chart	Виведення графіків зведених даних	Назви колонок, дата фрейм	Діаграма зведених даних
12	read_file	Зчитування файлу	Шлях до файлу з датасетом	Дата фрейм
13	build_boxplot	Виведення діаграми-розмаху	Дата фрейм, назва колонки	Діаграма-розмаху
14	out_predict_data	Запис результуючих даних у файл	Вхідні дані, модель алгоритму	Рядок з вихідними даними
15	build_histogram	Виведення гістограми	Назва колонки, датафрейм	Гістограма
16	choose_function	Обробка вибору операції	-	Повернення обраного вікна

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
17	data_prediction_rf	Навчання система за допомогою алгоритму Random Forest	Дані для навчання, тестові дані, очікувані тестові дані, очікувані дані для навчання	Рядок з результатами навчання
18	check_value_col	Перевірка значення колонок	Дата фрейм, назва колонки для помилок	Перевірені дані
19	check_entered_param	Перевірка правильності введених даних	Назви колонок у даних	False при помилок, True – при успіху
20	init_main	Завантаження головного вікна	-	Головне вікно
21	choose_chart	Обробка вибору побудови графіків	-	Не повертає значень

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
22	prediction	Пошук майбутніх даних на основі вхідних	Дата фрейм, список параметрів прогнозування назва колонки помилок, тип алгоритму	None у випадку помилки, інакше – прогнозовані дані
23	data_prediction_lr	Навчання система за допомогою алгоритму Logical Regression	Дані для навчання, тестові дані, очікувані тестові дані, очікувані дані навчання	Рядок з результатами навчання
24	init_functional_window	Виведення вікна для вибору операцій	-	Вікно з вибором операцій
25	enable_button	Обробка стану кнопки	-	Увімкнена кнопка чи ні

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
26	accuracy_error_prediction	Розрахунок помилок та похибок	Дані для тестування та навчання, очікувані вихідні дані після тестування та навчання, модель навчання	Вивід помилок та похибок у передбачених даних
27	open_dialog	Відкриття вікна для вибору файлу та перехід у інше вікно	-	У разі вибору файлу відкривається нове вікно, у іншому випадку – нічого не відбувається
28	menu_window	Додавання вкладки меню до головного вікна	-	Вкладка у вигляді вибору

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
29	init_data_window	Виведення вікна з даними	-	Вікно з датасетом
30	init_analyze_window	Виведення вікна для аналізу	-	Вікно для аналізу
31	exit_window	Вихід з поточного вікна	-	Повернення до батьківського вікна та закриття поточного. У випадку головного вікна – закриття програми
32	build_chart	Побудова графіків	Назва звичайної колонки, з помилками, id, номер операції, шлях до файлу	Побудований графік

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
33	init_result_window	Виведення вікна з результами	-	Вікно для відображення результатів
34	is_int	Перевірка на позитивне ціле число	Введене значення	True у випадку правильності введені, False – навпаки
35	init_param_analyze_window	Виведення вікна вікна для введення параметрів аналізу	-	Вікно для введення параметрів аналізу
36	init_predicted_window	Завантаження вікна для введення параметрів (передбачення даних)	-	Вікно для введення параметрів для передбачення даних
37	open_param_window	Виведення вікна для введення додаткових параметрів	-	Вікно для введення додаткових параметрів

4.1.2 Стандартні функції

У таблиці 4.2 наведені функції, які прописані у бібліотеках та використовуються для роботи системи.

Таблиця 4.2 – Стандартні функції

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	to_datetime	Змінити дату на рядок	Дата	Дата у вигляді рядка
2	read_csv	Зчитування csv-файлів	Шлях відносний до файлу	DataFrame
3	get_cmap	Отримання кольорів	Кількість кольорів	Масив кольорів
4	arange	Формування послідовності чисел	Розмір масиву	Сформований масив
5	sorted	Сортування даних	Словник (масив, список)	Відсортований словник (масив, список)
6	count	Повернення індексу списку	Значення, списку	Індекс елемента масиву(списку)
7	append	Додавання елементів в список (масив, словник)	Значення, які потрібно вставляти	Не повертає значення

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
8	len	Знаходження довжини масиву	Масив (список)	Довжина масиву (списку)
9	xlabel	Назва осі Ox	Назва осі	Не повертає значень
10	ylabel	Назва осі Oy	Назва осі	Опис вихідних параметрів
11	xticks	Позначення на осі Ox	Масив значень	Не повертає значень
12	yticks	Позначення на осі Oy	Масив значень	Не повертає значень
13	title	Позначення діаграми	Назва діаграми	Не повертає значень
14	RandomForestClassifier	Створення моделі для алгоритму RandomForest	Кількість дерев рішень	Модель для алгоритму RandomForest
15	LogisticRegression	Створення моделі для лінійної регресії	Кількість ітерацій	Модель для лінійної регресії
16	fit	Навчання моделі	Тренувальні дані, очікувані вихідні дані	Навчена модель
17	predict	Передбачення даних	Тестові дані	Передбачені дані

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	show	Виведення діаграми	-	Повертає діаграму
19	train_test_split	Розбиття даних для тестування та тренування	Відсоток тестових значень, датасет, усі значення колонки, значення для генератора випадкових значень датасету	Дані для тестування та навчання, очікувані вихідні дані після тестування та навчання
20	mean_absolute_error	Розрахунок середньої абсолютної помилки	Тестові дані, створена модель навчання	Середня абсолютна помилка
21	score	Розрахунок ймовірності правильності передбачених даних	Тестові дані, очікувані вихідні тестові дані	Відсоток правильності даних
22	info	Вивід інформації про датасет	Датасет	Вивід інформації про датасет

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
23	pivot_table	Зведення даних	Назва колонок	Дата фрейм зведених даних
24	mean_squared_error	Розрахунок середньої квадратичної помилки	Очікувані вихідні тестові дані, створена модель навчання	Середня квадратична помилка
25	describe	Розрахунок характеристик описової статистики	Датасет	Розраховані характеристик описової статистики
26	mean_squared_error	Розрахунок середньої квадратичної помилки	Числові значення	Середня квадратична помилка
27	sqrt	Розрахунок квадратного кореня	Числові значення	Результати функції
28	info	Вивід інформації про датасет	Датасет	Вивід інформації про датасет
29	describe	Розрахунок мір описової статистики	Датасет	Розраховані міри описової статистики

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
30	sqrt	Розрахунок квадратного кореня	Значення для розрахунку кореня	Результати функції
31	distplot	Відображає гістограму і діаграму щільності	Назву колонки для відтворення даних	Гістограма і діаграма щільності
32	bar	Відображення даних у вигляді діаграм	Значення по Ох та Оу	Діаграма
33	dump	Збереження моделі для навчання	Назва моделі, модель	Не повертає значень
34	boxplot	Побудова діаграми-розмаху	Назва колонки, для якої потрібно побудувати діаграму	Діаграма-розмаху
35	load	Завантаження моделі для навчання	Назва моделі	Бажана модель
36	protocol	Обробка натиснення на кнопку «Х» вікна	Назва події, функція на виклик події	Функція для обробки події

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
37	pack	Відображення віджетів	-	Віджети
38	askyesno	Виведення вікна з повідомленням	Рядок	Вікно з повідомленням
39	geometry	Розміщення вікна	Параметри для розміщення	Не повертає результат
40	Menu	Поява віджету «меню»	Поточне вікно	Віджет «menu»
41	resizable	Дозвіл (заборона) зміни розміру вікна	True або False	Не повертає результат
42	askopenfilename	Відкриття провідника для вибору файлу	Шлях для відкриття папки, назва файлу список розширення файлів	Завантаження файлу після вибору
43	focus_set	Встановлення фокусу на поточне вікно	-	Не повертає результат

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
44	grab_set	Заборона використання батьківського вікна	-	Не повертає результат
45	add_command	Заповнення віджету «меню»	Назва пункту меню, функція обробки	Не повертає значень
46	place	Встановлення положення віджету	Координати x та y	Не повертає результати
47	add_cascade	Встановлення пункту каскадного відображення «меню»	Назва пунктів меню	Віджет «menu»
48	Radiobutton	Поява віджету «кнопки з одним вибором»	Поточне вікно, назва кнопки, значення при виборі кнопки	Віджет «menu»
49	destroy	Закриття поточного вікна	-	Не повертає значень

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
50	grid	Встановлення положення віджетів у вигляді таблиці та відображення	Номери колонки, рядка	Відображення віджетів
51	Frame	Поява віджету «frame»	Поточне вікно	Відображення віджету
52	Label	Поява віджету «label»	Поточне вікно, значення віджету, параметри для зображення вікна	Відображення віджету
53	Text	Поява віджету «text»	Поточне вікно, будь-які параметри для зображення вікна	Не повертає результати
54	deiconify	Відновлення прихованого вікна	-	Не повертає результат

№ п/п	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
55	Entry	Поява віджету «entry»	Поточне вікно, поле та пов’язана змінна для введення даних	Відображення віджету
56	Table	Поява віджету «table»	Віджет «frame», параметри для розташування	Відображення таблиці
57	withdraw	Приховати вікно	-	Не повертає результат

Як видно з таблиць 4.1 та 4.2, розроблена система має наступний функціонал:

– дані зчитуються з файлу;

– для аналізу відбувається обчислення мір дескриптивної (описової) статистики;

– є можливість будувати графіки, діаграми, гістограми для аналізу даних;

– передбачення даних відбувається за допомогою машинного навчання, для якого можна обирати той чи інший алгоритм;

– результати роботи системи (модель навчання та прогнозовані дані) записується у файл з розширенням .pkl та .csv відповідно;

– присутня перевірка правильності навчання системи за допомогою обчислення похибок;

– для взаємодією з системою створений графічний інтерфейс, що містить різноманітні віджети: кнопки, текстові поля, випадаюче меню та інші. Це дає змогу більш зручно працювати;

– існує обробка помилок на випадок помилкового введення даних.

Для перевірки коректності даного функціоналу нижче описано тестування системи.

4.2 Тестування системи

Важливим етапом у розробці систем (програмного забезпечення) є тестування. Це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Існують дуже багато видів класифікацій. Ключовою є класифікація за ступенем автоматизації. Тут виділяють наступні види тестів:

– ручні – перевірка функціоналу самостійно;

– автоматизовані – за допомогою написаних тестів;

– напівавтоматизовані – з частковим використанням написаних тестів.

Для тестування розробленої системи використано спочатку ручне. Воно дало гарні результати, однак для повної перевірки правильності роботи системи, використано модульне тестування (Unit-test).

Unit-test – технічне дослідження програмного забезпечення, яке полягає в окремій перевірці кожного модуля коду програми.

Даний метод обраний, оскільки за допомогою нього є можливість протестувати в повній мірі написану систему.

За допомогою автоматизованих тестах перевіряються наступні частини системи:

– правильність зчитування даних з файлу;

– коректність заповнення пустих комірок у дата фреймі;
– правильність обчислення медіани;
– реакція функції на введення помилкових або правильних даних;
– вірність запису результатів прогнозування до файлів;
– повернення прогнозованих даних;
– правильність вихідних параметрів;

Результати виконання модульних тестів на рисунку 4.1.

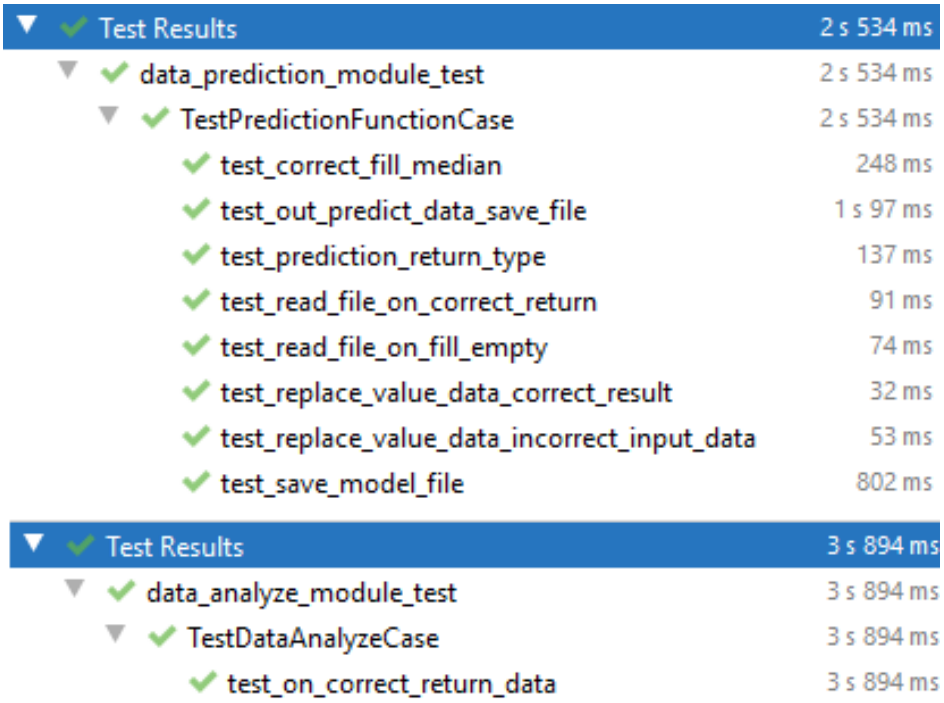


Рисунок 4.1 – Результати роботи модульних тестів для прогнозування даних та аналізу даних

Як видно з рисунків, тести успішно пройшли, а отже система працює вірно.

4.3 Опис роботи системи

Головною завданням даного проєкту є побудова автоматизованої системи пошуку несправностей в корпоративних IT-інфраструктурах, яку б в подальшому можна використовувати у різних сферах IT для покращення

стану автоматизації. Виходячи цього, даний розділ присвячений опису роботи системи.

4.3.1 Аналізування даних у системі

Після запуску системи з’являється привітальне вікно, у якому можна натиснути на кнопку у вигляді файлу як на рисунку 4.2.

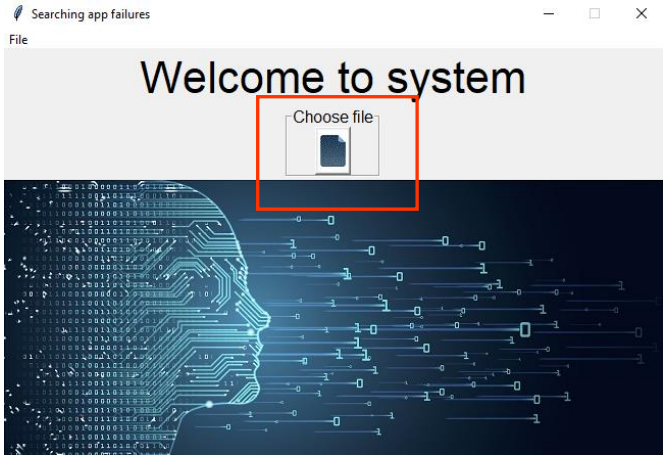


Рисунок 4.2 – Привітальне вікно системи

Натиснувши на кнопку, ми можемо обирати файл з розширенням .csv, оскільки інші файли система не дозволить обрати, та перейти у вікно вибору функції, яке зображено на рисунку 4.3.

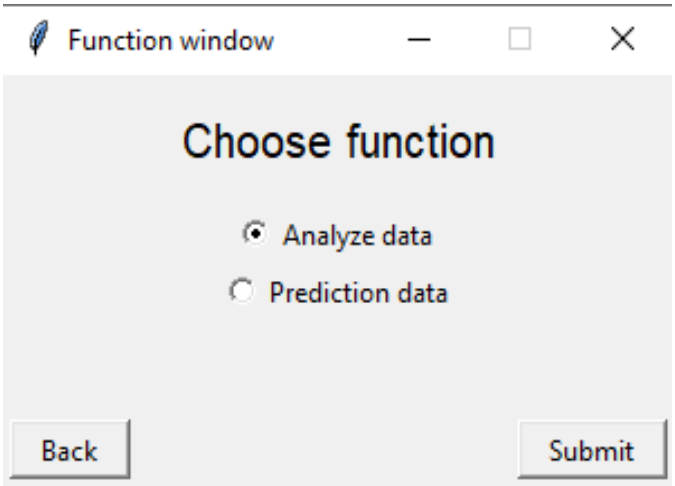
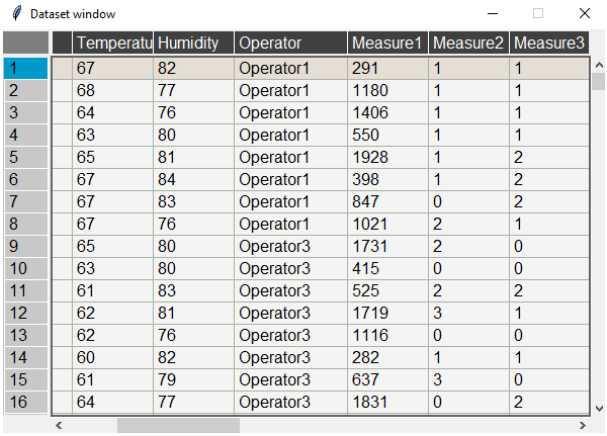


Рисунок 4.3 – Вікно вибору функції

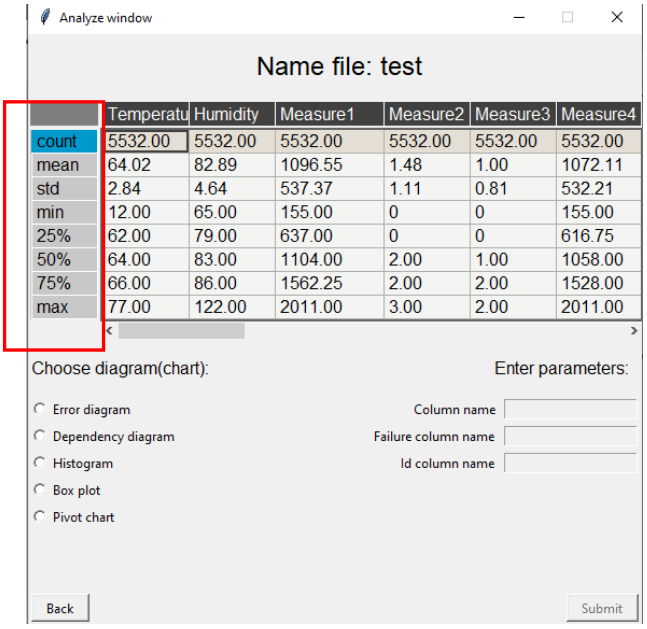
Також з’явиться вікно з відображенням файлу, який обрано. Дане вікно відображене на рисунку 4.4.



	Temperatu	Humidity	Operator	Measure1	Measure2	Measure3
1	67	82	Operator1	291	1	1
2	68	77	Operator1	1180	1	1
3	64	76	Operator1	1406	1	1
4	63	80	Operator1	550	1	1
5	65	81	Operator1	1928	1	2
6	67	84	Operator1	398	1	2
7	67	83	Operator1	847	0	2
8	67	76	Operator1	1021	2	1
9	65	80	Operator3	1731	2	0
10	63	80	Operator3	415	0	0
11	61	83	Operator3	525	2	2
12	62	81	Operator3	1719	3	1
13	62	76	Operator3	1116	0	0
14	60	82	Operator3	282	1	1
15	61	79	Operator3	637	3	0
16	64	77	Operator3	1831	0	2

Рисунок 4.4 – Приклад вікна з даними

Обравши аналіз даних, з’являється вікно, у якому можна натиснути на нумерацію рядків, вибрати пункт «Toggle index» і навпроти розрахованих значень з’являться назви статистичних мір. Приклад наведено на рисунку 4.5.



	Temperatu	Humidity	Measure1	Measure2	Measure3	Measure4
count	5532.00	5532.00	5532.00	5532.00	5532.00	5532.00
mean	64.02	82.89	1096.55	1.48	1.00	1072.11
std	2.84	4.64	537.37	1.11	0.81	532.21
min	12.00	65.00	155.00	0	0	155.00
25%	62.00	79.00	637.00	0	0	616.75
50%	64.00	83.00	1104.00	2.00	1.00	1058.00
75%	66.00	86.00	1562.25	2.00	2.00	1528.00
max	77.00	122.00	2011.00	3.00	2.00	2011.00

Рисунок 4.5 – Приклад вікна з розрахованими мірами описової статистики

З зображеного вікна на рисунку (див. Рисунок 4.5) видно, що є можливість будувати графіки, вказувавши, відповідні параметри.

Для побудови графіку потрібно обрати тип графіку, ввести параметри. Приклад заповнених даних наведено на рисунку 4.6.

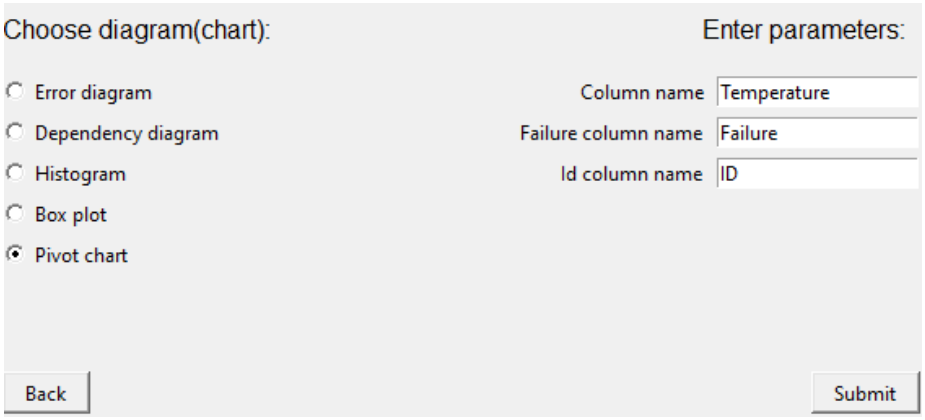


Рисунок 4.6 – Приклад вікна з заповненими даними для побудови графіку

Після підтвердження дії, відобразиться діаграма, яку обрано, як показано на рисунку 4.7.

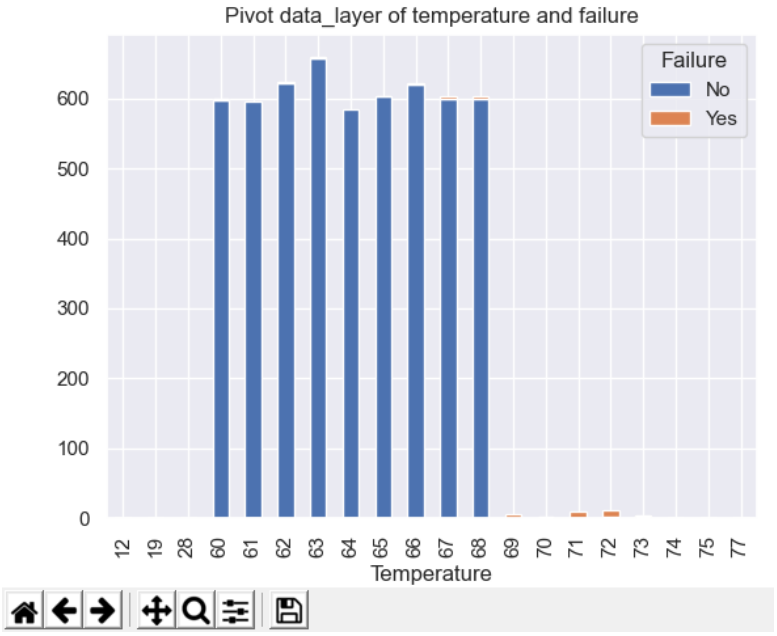


Рисунок 4.7 – Діаграма зведених даних

Аналогічним способом будуються й інші графіки чи діаграми.

Після вибору графіку система робить активними лише поля, які потрібні для цієї побудови, а усі інші залишаються неактивними.

Отримані графіки можна зберігати як картинку, збільшувати масштаб графіку та інші дії.

4.3.2 Прогнозування даних у системі

Важливим функціоналом даної системи є прогнозування даних. Для використання цієї функції потрібно обрати її у вікні вибору аналогічним способом, як описувалось у підпункті 4.6.1.

Після цього відкриється вікно передбачення, де потрібно вказати кількість колонок, для яких робити прогнозування, назва колонки, що описує виникнення помилок. Вказання назви колонки, що відповідає за несправність важливо, бо головна мета цієї системи знайти дані, за якими можливий збій у досліджуваної системи. Приклад даного вікна з заповненими даних зображено на рисунку 4.8.

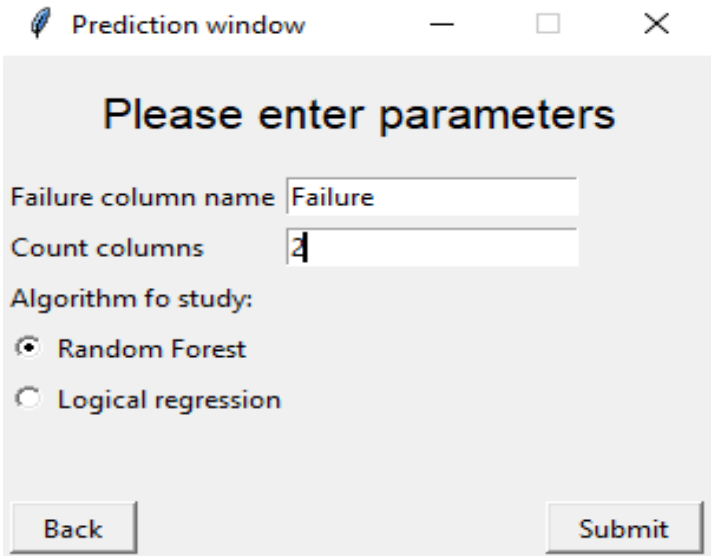


Рисунок 4.8 – Приклад вікна з заповненими даними

Натиснувши на підтвердження, відбувається перехід до додаткового вікна, яке зображено на рисунку 4.10.

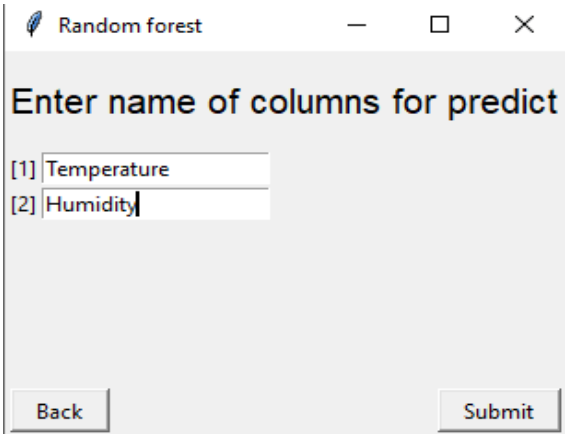


Рисунок 4.9 – Приклад додаткового вікна з заповненими вікнами

У результаті підтвердження з’являться два вікна:

– результуюче вікно містить дані, які вказуватимуть на кількість помилок у системі, які передбачені, розрахунки похибок навчання, а також повідомлення про успішність запису результату у файл. Вікно результату показано на рисунку 4.11.

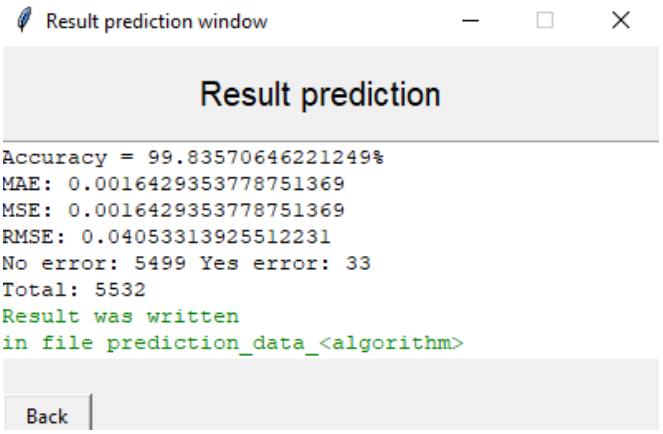


Рисунок 4.10 – Результуюче вікно для прогнозування даних

– вікно з виведенням вмісту файлу, у який записаний результат. Це зображено на рисунку 4.13.

Dataset window

	ID	Temperatu	Humidity	Predicted
1	1	67	82	0
2	2	68	77	0
3	3	64	76	0
4	4	63	80	0
5	5	65	81	0
6	6	67	84	0
7	7	67	83	0
8	8	67	76	0
9	9	65	80	0
10	10	63	80	0
11	11	61	83	0
12	12	62	81	0
13	13	62	76	0
14	14	60	82	0
15	15	61	79	0
16	16	64	77	0
17	17	60	80	0

Рисунок 4.11 – Відображення файлу з результатами прогнозування даних

Висновки до розділу 4

Даний розділ присвячений опису роботи системи, функціональної складової, а також її тестуванню.

Для першого пункту описано на прикладах роботу функцій для аналізу даних у даній системі (знаходження мір статистичних, побудова графіків), а також функцій для прогнозування даних.

Для тестування системи обрані ручні та модульні автоматизовані модулі. Обидва види дали гарні результати. Виходячи з цього, можна робити висновок, що система працює правильно та не дає помилкових результатів.

Згідно описаних результатів можна робити висновки, що система задовольняє усім сформованим вимогам та дає можливість використовувати для рішення поставленого завдання.

Код програмної реалізації даної системи, що вміщує реалізацію самої системи та її тестування знаходиться у додатку А – Код для роботи системи код, та Код для тестування системи відповідно.

ВИСНОВКИ

Мета даного дипломного проєкту – покращення рівня автоматизації в корпоративних IT-інфраструктурах.

Головне завдання – розробка автоматизованої системи пошуку несправностей (АСПН) в корпоративних IT-інфраструктурах.

Створення даної системи починалось з огляду та аналізу раніше створених проєктів:

– ELK-stack:

– Graylog 2:

– Splunk.

Були проаналізовані переваги та недоліки кожного з цих продуктів, технології, які вони використовують для вирішення задач, методи, алгоритми машинного навчання та інше.

Етап розробки вміщував у собі поставлення вимог до розроблюваної системи, опис сценарію роботи системи, механізм обробки запитів користувача, а також структурне представлення системи. Для цього знадобились методології IDEF0 та IDEF3, use-case діаграми та інші.

Виходячи з аналізу існуючих рішень та вимог до системи, були обрані теоретичні метод та програмні інструменти для реалізації АСПН. Серед теоретичних методів – розрахунок статистичних мір, обраний алгоритми з категорії «навчання з вчителем»: «Випадковий ліс» (Random Forest») та логічну регресію, розрахунок похибок: середньої абсолютної, середньої квадратичної та інших. Серед програмних інструментів обрана мова програмування Python, бібліотека для машинного навчання Scikit-learn, бібліотеки для аналізу даних: pandas, matplotlib, seaborn, бібліотека для інженерних розрахунків numpy. Для графічного представлення використана бібліотека Tk та транслятор Tkinter с мови програмування Python на Tsl.

Для більш комфортної розробки на початковому етапі використовувалось середовище Jupyter Lab з дистрибутиву Anaconda, для

повної реалізації системи – PyCharm, для контролю версій – Git та сайт для хостингу GitHub.

У результаті усіх описаних дій, побудована система, яка відповідає поставленим вимогам. Для перевірки її проведено ручне тестування, а також модульне тестування, що дали гарний результат.

Створена система, може використовуватись у IT-сфері, як частина іншої системи, оскільки вона поки не містить досить великий функціонал. Щоб використовувати лише цю систему, її потрібно покращувати. Зокрема, написати функціонал, який дозволив би не тільки аналізувати логи та прогнозувати дані у якості комп'ютерної версії, але й для веб-версії, наприклад. Також, базуючись на вже написаному функціоналі, можна додати у систему можливість автоматичного збору даних з пристроїв без використання аналітиків, самій шукати аномалії, прогнозувати можливі несправності та повідомляти про ці помилки за певний час до їх виникнення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ролик А.И. Управление корпоративной ИТ-инфраструктурой / А.И. Ролик, С.Ф. Теленик, М.В. Ясочка // К.: Наукова думка, 2018. – 576 с. (https://acts.kpi.ua/app/uploads/2020/05/rolik_teleni_yasochka_uiti.pdf).
2. What is the ELK Stack? URL: https://www.elastic.co/what-is/elk-stack (Last accessed: 28.02.2020).
3. Elasticsearch URL: https://ru.wikipedia.org/wiki/Elasticsearch (Last accessed: 02.03.2020).
4. Elastic Logstash URL: https://ru.bmstu.wiki/Elastic_Logstash (Last accessed: 04.03.2020).
5. Что такое стандартное отклонение URL: https://exceltip.ru/что-такое-стандартное-отклонение-исп/ (Дата обращения: 10.03.2020).
6. Машинне навчання URL: https://uk.wikipedia.org/wiki/Машинне_навчання (Дата звернення: 11.03.2020).
7. Обучение нейросети с учителем, без учителя, с подкреплением – в чем отличие? Какой алгоритм лучше URL: https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/ (Дата звернення: 12.03.2020).
8. Обзор самых популярных алгоритмов машинного обучения URL: https://tproger.ru/translations/top-machine-learning-algorithms/ (Дата обращения: 17.03.2020).
9. What is Mean Squared Error, Mean Absolute Error, Root Squared Error and R Squared? URL: https://www.studytonight.com/post/what-is-mean-squared-error-mean-absolute-error-root-mean-squared-error-and-r-squared (Last accessed: 21.03.2020).
10. Python URL: https://uk.wikipedia.org/wiki/Python#Функціональне_програмування (Last accessed: 27.03.2020).

11. 5 лучших библиотек машинного обучения URL: https://medium.com/nuances-of-programming/nuances-of-programming/5-лучших-библиотек-машинного-обучения-f8b6ddf50945 (Дата обращения: 03.04.2020).

12. Библиотеки Python, необходимые для машинного обучения URL: https://techrocks.ru/2018/10/05/python-libraries-for-machine-learning/ (Дата обращения: 13.04.2020).

13. Pandas (software) URL: https://en.wikipedia.org/wiki/Pandas_(software) (Last accessed: 14.04.2020).

14. Matplotlib URL: https://en.wikipedia.org/wiki/Matplotlib (Last accessed: 15.04.2020).

15. Getting Started with Plotly in Python URL: https://plotly.com/python/getting-started/?utm_source=mailchimp-jan-2015&utm_medium=email&utm_campaign=generalemail-jan2015&utm_term=bubble-chart (Last accessed: 20.04.2020).

16. Tkinter – Python interface to Tcl/Tk URL: https://docs.python.org/3/library/tkinter.html (Last accessed: 27.04.2020).

ДОДАТОК А

Програмний код

Код для роботи системи

main.py

```
import troubleshooting_system.ui_layer.start_window as ww
from tkinter import *
from PIL import ImageTk, Image

def call_window(window):
    app = ww.WelcomeWindow(window)
    app.pack()
if __name__ == "__main__":
    root = Tk()
    call_window(root)
    canvas = Canvas(root, width=650, height=450)
    img = ImageTk.PhotoImage(Image.open('E:\\Project\\Automated-troubleshooting-system\\troubleshooting_system'\\images_layer\\background.jpg'))
    canvas.create_image(0, 0, anchor=NW, image=img)
    canvas.pack()
    root.mainloop()
```

accuaracy_module.py

```
from sklearn import metrics
import numpy as np

def accuracy_error_prediction(inputs_test, expected_output_test, predicted, algorithm):
    accuracy = algorithm.score(inputs_test, expected_output_test)
    return "Accuracy = {}%".format(accuracy * 100) + "\n" + \
    "MAE: " + str(metrics.mean_absolute_error(expected_output_test,predicted)) + "\n"\ "MSE: " + str(metrics.mean_squared_error(expected_output_test, predicted)) + "\n" + \ "RMSE: " + str(np.sqrt(metrics.mean_squared_error(expected_output_test, predicted))) + "\n"
```

analyze_module.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from troubleshooting_system.data_analyze_layer.build_chart_module import get_colors
from troubleshooting_system.data_analyze_layer.data_processing_module import
data_error, dictionary_sort

pd.options.mode.chained_assignment = None
PLOT_LABEL_FONT_SIZE = 14

def find_statistics_param(file):
    data = pd.read_csv(file)
    try:
        data.Date = data.Date.apply(pd.to_datetime)
        data = data.drop(['ID'], axis=1)
    except AttributeError:
        return data.describe()
    except KeyError:
        return data.describe()
    return data.describe()

def build_dependency_diagram(name_column, failure_column, data):
    sns.set()
    plt.close("Dependency diagram (line)")
    plt.close("Dependency diagram")
    diagram_param = data_error(name_column, failure_column, data)
    plt.figure("Dependency diagram")
    plt.xlabel(name_column, fontsize=PLOT_LABEL_FONT_SIZE)
    plt.ylabel('Count of failures', fontsize=PLOT_LABEL_FONT_SIZE)

    plt.bar(np.arange(diagram_param[0]), diagram_param[1],
color=get_colors(diagram_param[0]))
    plt.xticks(np.arange(diagram_param[0]), diagram_param[2], rotation=0,
fontsize=12)
    plt.yticks(fontsize=PLOT_LABEL_FONT_SIZE)
    name_column_diagram = name_column.lower()
    plt.title('Dependency between failure and ' + name_column_diagram,
fontsize=PLOT_LABEL_FONT_SIZE)
    plt.figure("Dependency diagram (line)")
```

```
    plt.xlabel(name_column, fontsize=PLOT_LABEL_FONT_SIZE)
    plt.ylabel('Count of failures', fontsize=PLOT_LABEL_FONT_SIZE)
    plt.title('Dependency between failure and ' + name_column_diagram,
fontsize=PLOT_LABEL_FONT_SIZE)
    plt.plot(diagram_param[2], diagram_param[1])
    plt.show()

def build_pivot_chart(name_column, failure_column, data, id_col):
    sns.set()
    data.pivot_table(id_col, name_column, failure_column, 'count'). \
        plot(kind='bar', stacked=True,
             title="Pivot data_layer of " + name_column.lower() + " and " +
failure_column.lower())
    plt.show()

def build_histogram(data, name_column):
    sns.set()
    plt.figure("Histogram")
    plt.title("Histogram " + name_column.lower())
    sns.distplot(data[name_column], color='g', bins=100, hist_kws={'alpha': 0.4})
    plt.show()

def build_boxplot(data, name_column):
    plt.figure("Box plot")
    plt.title("Box plot " + name_column.lower())
    sns.boxplot(x=data[name_column])
    plt.show()
def build_error_diagram(data, failure_column):
    sns.set()
    failure_count = pd.value_counts(data[failure_column].values, sort=True)
    failure_count_keys, failure_count_values = dictionary_sort(dict(failure_count))
    failures = len(failure_count_keys)
    plt.figure("Error chart")
    plt.title(failure_column + ' ', fontsize=PLOT_LABEL_FONT_SIZE)
    plt.bar(np.arange(failures), failure_count_values)
    plt.xticks(np.arange(failures), failure_count_keys, rotation=0, fontsize=12)
    plt.yticks(fontsize=PLOT_LABEL_FONT_SIZE)
    plt.ylabel('Count of failures', fontsize=14)
    plt.show()
```

build_chart_module.py

```
import troubleshooting_system.data_analyze_layer.analyze_module as adm
import troubleshooting_system.data_analyze_layer.data_processing_module as dpm
import matplotlib.pyplot as plt
import numpy as np

def build_chart(param_col, param_fail, param_id, param, file):
    if param == 5:
        try:
            adm.build_pivot_chart(param_col, param_fail, dpm.read_file(file),
param_id)
        except KeyError:
            return False
    if param == 1:
        try:
            adm.build_error_diagram(dpm.read_file(file), param_fail)
        except KeyError:
            return False
    if param == 2:
        try:
            adm.build_dependency_diagram(param_col, param_fail, dpm.read_file(file))
        except KeyError:
            return False
    if param == 3:
        try:
            adm.build_histogram(dpm.read_file(file), param_col)
        except KeyError:
            return False
    if param == 4:
        try:
            adm.build_boxplot(dpm.read_file(file), param_col)
        except KeyError:
            return False
    return True
```

```
def get_colors(count):
    colors = []
    cm = plt.cm.get_cmap('hsv', count)
    for i in np.arange(count):
        colors.append(cm(i))
    return colors
```

data_processing_module.py

```
import pandas as pd
from troubleshooting_system.data_analyze_layer.validation_module import
check_value_col

def read_file_prediction(file):
    data = pd.read_csv(file, index_col=0)
    for col in data.columns:
        if data[col].dtype == object:
            continue
        median = data[col].median()
        data[col].fillna(median, inplace=True)
    return data

def read_file(file):
    data = pd.read_csv(file)
    return data

def dictionary_sort(my_dict):
    keys = []
    values = []
    my_dict = sorted(my_dict.items(), key=lambda x: x[1], reverse=True)
    for key, value in my_dict:
        keys.append(key)
        values.append(value)
    return keys, values

def replace_value_data(data, lst_param, fail_param):
    try:
        if check_value_col(data, fail_param) is None:
            raise Exception()
```

```
            expected_output = check_value_col(data, fail_param)
            data_inputs = data[lst_param]
            return expected_output, data_inputs
        except KeyError:
            return None
        except Exception:
            return None

def data_error(name_column, failure_column, data):
    array = []
    count_array = []
    data_set = set()
    for row in data.iterrows():
        if row[1][failure_column] == 'Yes':
            array.append(row[1][name_column])
        if row[1][failure_column] == 1:
            array.append(row[1][name_column])
    array.sort()
    for element in array:
        data_set.add(element)
    data_set = sorted(data_set, key=None, reverse=False)

    for element_set in data_set:
        count_value = array.count(element_set)
        count_array.append(count_value)
    count_value = len(count_array)
    return count_value, count_array, data_set
```

prediction_data.py

```
import uuid
import joblib
import pandas as pd
from troubleshooting_system.data_analyze_layer.accuracy_module import
accuracy_error_prediction
from troubleshooting_system.data_analyze_layer.data_processing_module import
replace_value_data
from troubleshooting_system.data_analyze_layer.save_module import out_predict_data
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
from pathlib import Path
pd.options.mode.chained_assignment = None

def prediction(data, lst_param, fail_param, algorithm):
    changed_data = replace_value_data(data, lst_param, fail_param)

    if changed_data is None:
        return None

    inputs_train, inputs_test, expected_output_train, expected_output_test =
train_test_split(changed_data[1],

changed_data[0],

test_size=0.33,

random_state=42)
    if algorithm == 1:
        return data_prediction_rf(inputs_train, inputs_test, expected_output_train,
expected_output_test,
                                  changed_data[1])
    else:
        return data_prediction_lr(inputs_train, inputs_test, expected_output_train,
expected_output_test,
                                  changed_data[1])


def data_prediction_rf(inputs_train, inputs_test, expected_output_train,
expected_output_test, data_inputs):
    rf = RandomForestClassifier(n_estimators=100)
    rf.fit(inputs_train, expected_output_train)
    predicted = rf.predict(data_inputs)
    predict_for_test = rf.predict(inputs_test)
    return str(accuracy_error_prediction(inputs_test, expected_output_test,
predict_for_test, rf)) + \
           str(out_predict_data(predicted, data_inputs, 'r', rf))
def data_prediction_lr(inputs_train, inputs_test, expected_output_train,
expected_output_test, data_inputs):
    lr = LogisticRegression(max_iter=8000)
    lr.fit(inputs_train, expected_output_train)
```

```
    predicted = lr.predict(data_inputs)
    home = str(Path.home())
    joblib.dump(lr, home + "\\prediction_data_lr_" + str(uuid.uuid4())+".pkl",
compress=9)
    predict_for_test = lr.predict(inputs_test)
    return str(accuracy_error_prediction(inputs_test, expected_output_test,
predict_for_test, lr)) + \
           str(out_predict_data(predicted, data_inputs, 'l', lr))
```

save_module.py

```
import uuid
import joblib
from pathlib import Path

def out_predict_data(predicted, data_inputs, flag, algorithm):
    count_no = 0

    count_yes = 0
    data_inputs['Predicted failure'] = predicted
    home = str(Path.home())
    if flag == 'l':
        data_inputs.to_csv(home + "\\prediction_data_lr.csv")
        joblib.dump(algorithm, str(Path.home()) + "\\prediction_data_lr_" +
str(uuid.uuid4()) + ".pkl", compress=9)
    else:
        data_inputs.to_csv(home + "\\prediction_data_rf.csv")
        joblib.dump(algorithm, str(Path.home()) + "\\prediction_data_rf_" +
str(uuid.uuid4()) + ".pkl", compress=9)
    for element in predicted:
        if element == 1:
            count_yes += 1
        else:
            count_no += 1
    return "No error: " + str(count_no) + " " + "Yes error: " + str(count_yes) + "\n"
+ \
           "Total: " + str(count_no + count_yes)
```

validation_module.py

```
import math
import numpy as np

def check_value_col(data, fail_param):
    try:
        expected_output = data[fail_param]
        if data[fail_param].dtype == object:
            count_flag = 0
            count_nan = 0
            for el in expected_output:
                if el == "No" or el == "Yes":
                    count_flag += 1
                elif type(el) != str:
                    nan = float(el)
                    if math.isnan(nan):
                        count_nan += 1
divider = expected_output.size - count_nan - count_flag
            if divider != 0:
                raise Exception
            expected_output = np.where(expected_output == "No", 0, 1)
            data[fail_param] = expected_output
            median = data[fail_param].median()
            data[fail_param].fillna(median, inplace=True)
            expected_output = data[fail_param]
        return expected_output
    except Exception:
        return None
```

analyze_window.py

```
import os
import tkinter
import troubleshooting_system.data_analyze_layer.analyze_module as ad
from tkinter import *
from tkinter import messagebox
from pandastable import Table
import troubleshooting_system.data_analyze_layer.build_chart_module as cw


class AnalyzeWindow(tkinter.Toplevel):
```

```
    def __init__(self, root, file):
        super().__init__(root)
        self.root = root
        self.param = IntVar()
        self.file = file
        self.flag_id = StringVar()
        self.flag_id.set(DISABLED)
        self.flag_col = StringVar()
        self.flag_col.set(DISABLED)
        self.flag_fail = StringVar()
        self.flag_fail.set(DISABLED)
        self.flag_btn = StringVar()
        self.flag_btn.set(DISABLED)
        self.col_name = StringVar()
        self.fail_col_name = StringVar()
        self.col_id_name = StringVar()
        self.init_analyze_window(file)
        self.param_col_entry = Entry(self, textvariable=self.col_name,
state=self.flag_col.get())
        self.param_fail_entry = Entry(self, textvariable=self.fail_col_name,
state=self.flag_fail.get())
        self.param_id_entry = Entry(self, textvariable=self.col_id_name,
state=self.flag_id.get())
        self.btn_submit = tkinter.Button(self, text="Submit", anchor=SW, padx=10,
command=self.open_window_chart
                                         , state=self.flag_btn.get())
        self.btn_submit.place(x=500, y=520)
        self.param_col_entry.grid(row=3, column=0, sticky=W, padx=442)
        self.param_fail_entry.grid(row=4, column=0, sticky=W, padx=442)
        self.param_id_entry.grid(row=5, column=0, sticky=W, padx=442)

        Label(self, text="Column name").grid(row=3, column=0, sticky=W, padx=355)
        Label(self, text="Failure column name").grid(row=4, column=0, sticky=W,
padx=318)
        Label(self, text="Id column name").grid(row=5, column=0, sticky=W, padx=343)
        self.protocol("WM_DELETE_WINDOW", self.exit_window)

    def init_analyze_window(self, file):
        self.title("Analyze window")
        self.geometry("570x550+300+80")
        self.resizable(False, False)
```

```
        self.param.set(0)
        frame = tkinter.Frame(self)
        pt = Table(frame, dataframe=ad.find_statistics_param(file), height=176)
        pt.show()

        file = os.path.splitext(os.path.basename(file))[0]
        file_label = Label(self, text="Name file: " + file, font="Arial 18", pady=15,
padx=210)
        file_label.grid(row=0, column=0, sticky=W)

        frame.grid(row=1, column=0, sticky=W)

        btn_back = tkinter.Button(self, text="Back", command=self.exit_window,
anchor=SW, padx=10)
        btn_back.place(x=3, y=520)

        chart_label = Label(self, text="Choose diagram(chart):", font="Arial 12",
pady=15)
        chart_label.grid(row=2, column=0, sticky=W)

        param_label = Label(self, text="Enter parameters:", font="Arial 12", pady=15)
        param_label.grid(row=2, column=0, sticky=W, padx=430)

        pivot_chart = Radiobutton(self, text="Pivot chart", variable=self.param,
value=5, command=self.choose_chart)
        err_diagram = Radiobutton(self, text="Error diagram", variable=self.param,
value=1, command=self.choose_chart)
        depend_diagram = Radiobutton(self, text="Dependency diagram",
variable=self.param,
                                     value=2, command=self.choose_chart)
        histogram = Radiobutton(self, text="Histogram", variable=self.param, value=3,
command=self.choose_chart)
        box_plot = Radiobutton(self, text="Box plot", variable=self.param, value=4,
command=self.choose_chart)
        err_diagram.grid(row=3, column=0, sticky=W)
        depend_diagram.grid(row=4, column=0, sticky=W)
        histogram.grid(row=5, column=0, sticky=W)
        box_plot.grid(row=6, column=0, sticky=W)
        pivot_chart.grid(row=7, column=0, sticky=W)
```

```
    def choose_chart(self):
        if self.param.get() == 2:
            self.flag_fail.set(NORMAL)
            self.param_fail_entry['state'] = self.flag_fail.get()
            self.flag_col.set(NORMAL)
            self.param_col_entry['state'] = self.flag_col.get()
            self.flag_id.set(DISABLED)
            self.param_id_entry['state'] = self.flag_id.get()
            self.btn_submit['state'] = NORMAL
        elif self.param.get() == 1:
            self.flag_fail.set(NORMAL)
            self.param_fail_entry['state'] = self.flag_fail.get()
            self.flag_col.set(DISABLED)
            self.param_col_entry['state'] = self.flag_col.get()
            self.btn_submit['state'] = NORMAL
        elif self.param.get() == 5:
            self.flag_fail.set(NORMAL)
            self.param_fail_entry['state'] = self.flag_fail.get()


            self.flag_col.set(NORMAL)
            self.param_col_entry['state'] = self.flag_col.get()
            self.flag_id.set(NORMAL)
            self.param_id_entry['state'] = self.flag_id.get()
            self.btn_submit['state'] = NORMAL
        else:
            self.flag_fail.set(DISABLED)
            self.param_fail_entry['state'] = self.flag_fail.get()

self.flag_col.set(NORMAL)
self.param_col_entry['state'] = self.flag_col.get()
            self.flag_id.set(DISABLED)
            self.param_id_entry['state'] = self.flag_id.get()
            self.btn_submit['state'] = NORMAL

    def open_window_chart(self):
        if self.check_entered_param(self.col_name.get(), self.fail_col_name.get(),
                                    self.col_id_name.get(), self.param.get()):
            if not cw.build_chart(self.col_name.get(), self.fail_col_name.get(),
self.col_id_name.get(),
```

```
                                    self.param.get(), self.file):
                messagebox.showerror(title="Error", message="Please enter correct
parameters")

 else:
            messagebox.showerror(title="Error", message="Please enter parameters")

    def check_entered_param(self, param_col, param_fail, param_id, param):
        if param == 1:
            if not param_fail.strip():
                return False
        elif param == 2:
            if not param_fail.strip() or not param_col.strip():
                return False
        elif param == 5:
            if not param_fail.strip() or not param_col.strip() or not
param_id.strip():
                return False
        else:
            if not param_col.strip():
                return False
        return True
def exit_window(self):
        self.quit()
        self.destroy()
        self.root.deiconify()
```

choose_function_window.py

```
import tkinter
from tkinter import *
import troubleshooting_system.ui_layer.analyze_window as aw
import troubleshooting_system.ui_layer.start_prediction_window as pw

class FunctionWindow(tkinter.Toplevel):
    def __init__(self, root, file):
        super().__init__(root)
        self.var = IntVar()
        self.root = root
```

```
        self.file = file
        self.init_functional_window()
        self.protocol("WM_DELETE_WINDOW", self.exit_window)

 def init_functional_window(self):
        self.title("Function window")
        self.geometry("300x180+300+200")
        self.focus_get()
        self.resizable(False, False)
        self.var.set(0)
        self.root.deiconify()
        function_label = Label(self, text="Choose function", font="Arial 15",
pady=15)
        function_label.pack()
        btn_back = tkinter.Button(self, text="Back", command=self.exit_window,
anchor=SW, padx=10)
        btn_submit = tkinter.Button(self, text="Submit",
command=self.choose_function, anchor=SW, padx=10)
        btn_submit.place(x=230, y=150)
        btn_back.place(x=3, y=150)
        analyze = Radiobutton(self, text="Analyze data", variable=self.var, value=0)
        predict = Radiobutton(self, text="Prediction data", variable=self.var,
value=1)
        analyze.pack()
        predict.pack()

    def choose_function(self):
        if self.var.get() == 0:
            aw.AnalyzeWindow(self, self.file)
            self.withdraw()
        elif self.var.get() == 1:
            pw.PredictionWindow(self, self.file)
            self.withdraw()

    def exit_window(self):
        self.destroy()
```

param_prediction_window.py

```
import tkinter
```

```
from tkinter import messagebox

import troubleshooting_system.ui_layer.result_prediction_window as rpw
import troubleshooting_system.data_analyze_layer.prediction_module as pd
import troubleshooting_system.data_analyze_layer.data_processing_module as dpm
from tkinter import *

class ParamPredictionWindow(tkinter.Toplevel):
    def __init__(self, root, file, algorithm, count, fail_col_name):
        super().__init__(root)
        self.file = file
        self.lst = []
        self.root = root
        self.algorithm = algorithm
        self.fail_col_name = fail_col_name

        self.init_param_analyze_window()
        for i in range(int(count)):
            self.lst.append(StringVar())
        self.labels = []
        self.entries = []
        for i in range(int(count)):
            self.labels.append(Label(self, text="[" + str(i + 1) + "]"))
            self.labels[-1].grid(row=i + 1, column=0, sticky=W)
            self.entries.append(Entry(self, textvariable=self.lst[i]))
            self.entries[-1].grid(row=i + 1, column=0, sticky=W, padx=20)
        if len(self.lst) > 0:
            btn_submit = tkinter.Button(self, text="Submit", command=self.prediction,
anchor=SW,
padx=10)
            btn_submit.place(x=235, y=200)
        self.protocol("WM_DELETE_WINDOW", self.exit_window)

    def init_param_analyze_window(self):
        if self.algorithm == 1:
            self.title("Random forest")
        else:
            self.title("Logical regression")
        self.grab_set()
        self.focus_set()
```

```
        self.geometry("305x230+300+200")

        btn_back = tkinter.Button(self, text="Back", command=self.exit_window,
anchor=SW, padx=10)
        btn_back.place(x=3, y=200)

        title_label = Label(self, text="Enter name of columns for predict",
font="Arial 15")
        title_label.grid(row=0, column=0, sticky=W, pady=15)

    def prediction(self):
        params = []
        for el in self.lst:
            params.append(el.get())
        try:
            flag = pd.prediction(dpm.read_file_prediction(self.file), params,
self.fail_col_name, self.algorithm)
            if flag is None:
                raise Exception("Please, enter correct name of column")
            rpw.ResultPredictionWindow(self, dpm.read_file_prediction(self.file),
params, self.fail_col_name,
                                       self.algorithm, self.file)
            self.withdraw()
        except Exception as e:
            messagebox.showerror(title="Error", message=e)

    def exit_window(self):
        self.destroy()
        self.root.deiconify()
        self.resizable(False, False)
```

result_prediction_window.py

```
import tkinter
from pathlib import Path
from tkinter import *
        self.geometry("600x400+300+200")

import troubleshooting_system.data_analyze_layer.prediction_module as pd
import troubleshooting_system.ui_layer.show_dataset_window as dw
```

```
class ResultPredictionWindow(tkinter.Toplevel):
    def __init__(self, root, data, params, fail_col_name, algorithm, file):
        super().__init__(root)
        self.var = IntVar()
        self.root = root
        self.data = data
        self.params = []
        self.params = params
        self.file = file
        self.fail_col_name = fail_col_name
        self.algorithm = algorithm
        self.init_result_window()

self.protocol("WM_DELETE_WINDOW", self.exit_window)

    def init_result_window(self):
        self.title("Result prediction window")
        self.geometry("400x240+300+200")
        self.focus_get()
        self.resizable(False, False)
        self.var.set(0)

        result_label = Label(self, text="Result prediction", font="Arial 15",
pady=15, padx=120)
        result_label.grid(row=0, column=0, sticky=W)

        btn_back = tkinter.Button(self, text="Back", command=self.exit_window,
anchor=SW, padx=10)
        btn_back.place(x=3, y=210)

        success_label = Label(text="\nResult was written\nin file
prediction_data_<algorithm>")
        text = Text(self, width=400, height=8)
        text.insert(1.0, pd.prediction(self.data, self.params, self.fail_col_name,
self.algorithm))
        text.tag_config('info', foreground="green")
        text.insert(END, success_label['text'], 'info')
        text.configure(state=DISABLED)
        text.grid(row=1, column=0, sticky=W)
        home = str(Path.home())
```

```
        if self.algorithm == 2:
            dw.DataWindow(self.root, home + "\\prediction_data_lr.csv")
        else:
            dw.DataWindow(self.root, home + "\\prediction_data_rf.csv")

    def exit_window(self):
        self.destroy()
        self.root.deiconify()
```

show_dataset_window.py

```
from tkinter import messagebox

from pandastable import Table
import tkinter
from tkinter import *
import troubleshooting_system.data_analyze_layer.data_processing_module as dpm

class DataWindow(tkinter.Toplevel):
    def __init__(self, root, file):
        super().__init__(root)
        self.var = IntVar()
        self.root = root
        self.file = file
        self.init_data_window()
        self.protocol("WM_DELETE_WINDOW", self.exit_window)

    def init_data_window(self):
        self.title("Dataset window")
self.var.set(0)

        frame = tkinter.Frame(self)
        pt = Table(frame, dataframe=dpm.read_file(self.file), height=400)
        pt.show()
        frame.pack(fill='both')

    def exit_window(self):
        self.destroy()
```

start_prediction_window.py

```
import tkinter
from tkinter import *
from tkinter import messagebox

import troubleshooting_system.data_analyze_layer.data_processing_module as dpm
import troubleshooting_system.data_analyze_layer.validation_module as vm
import troubleshooting_system.ui_layer.param_prediction_window as ppw

class PredictionWindow(tkinter.Toplevel):
    def __init__(self, root, file):
        super().__init__(root)
        self.algorithm = IntVar()
        self.root = root
        self.file = file
        self.count_data = StringVar()
        self.fail_col_name = StringVar()
        self.init_predicted_window()
        self.btn_submit = tkinter.Button(self, text="Submit",
command=self.open_param_window, anchor=SW, padx=10,
                                         state=DISABLED)
        self.btn_submit.place(x=230, y=220)
        self.protocol("WM_DELETE_WINDOW", self.exit_window)

    def init_predicted_window(self):
        self.title("Prediction window")
        self.geometry("300x250+300+200")
        self.resizable(False, False)
        self.algorithm.set(0)

        title_label = Label(self, text="Please enter parameters", font="Arial 15",
pady=15, padx=40)
        title_label.grid(row=1, column=0, sticky=W)

from tkinter import *
Label(self, text="Failure column name").grid(row=2, column=0, sticky=W, pady=2)
        Label(self, text="Count columns").grid(row=3, column=0, sticky=W, pady=2)
        param_fail_entry = Entry(self, textvariable=self.fail_col_name)
        param_count_data = Entry(self, textvariable=self.count_data)
```

```
 param_fail_entry.grid(row=2, column=0, sticky=W, padx=120)
        param_count_data.grid(row=3, column=0, sticky=W, padx=120)

        btn_back = tkinter.Button(self, text="Back", command=self.exit_window,
anchor=SW, padx=10)
        btn_back.place(x=3, y=220)
        algorithm_label = Label(self, text="Algorithm fo study:", pady=3)
        algorithm_label.grid(row=4, column=0, sticky=W)

rf = Radiobutton(self, text="Random Forest", variable=self.algorithm, value=1,
command=self.enable_button)
        lr = Radiobutton(self, text="Logical regression", variable=self.algorithm,
value=2, command=self.enable_button)
        rf.grid(row=5, column=0, sticky=W)
        lr.grid(row=6, column=0, sticky=W)

    def enable_button(self):
        if self.algorithm.get() >= 1:
            self.btn_submit['state'] = NORMAL

    def open_param_window(self):
        if not self.fail_col_name.get().strip():
            messagebox.showerror(title="Error", message="Column must not be empty")
        try:
            var = dpm.read_file_prediction(self.file)[self.fail_col_name.get()]
            integer = int(self.count_data.get())
            if not self.is_int(integer):
                raise Exception("Count must be an positive integer")
            if integer > 6 or integer == 0:
                messagebox.showerror(title="Error", message="Count must be less than
7")
            elif not self.fail_col_name.get().strip():
                messagebox.showerror(title="Error", message="Column must not be
empty")
            flag = vm.check_value_col(dpm.read_file_prediction(self.file),
self.fail_col_name.get())
            if flag is None:
                raise Exception("Values for Failure column must be 0/1 (No/Yes)")
            ppw.ParamPredictionWindow(self, self.file, self.algorithm.get(),
self.count_data.get(),
```

```
                                        self.fail_col_name.get())
            self.withdraw()
        except KeyError:
            messagebox.showerror(title="Error", message="Please enter correct name of
column")
        except ValueError:
            messagebox.showerror(title="Error", message="Count must be an positive
integer")
        except Exception as e:
            messagebox.showerror(title="Error", message=e)

    def is_int(self, value):
        try:
            var = float(value)
            if var.is_integer():
                if var <= 0:

                    return False
                else:
                    return True
        except ValueError:
            return False

    def exit_window(self):
        self.destroy()
        self.root.deiconify()
```

start_window.py

```
import tkinter
import troubleshooting_system.ui_layer.choose_function_window as cfw
import troubleshooting_system.ui_layer.show_dataset_window as dw
from tkinter import filedialog, messagebox
class WelcomeWindow(tkinter.Frame):
    def __init__(self, root):
        super().__init__(root)
        self.icon = tkinter.PhotoImage(file='E:\\Project\\Automated-troubleshooting-
system\\'
'troubleshooting_system\\images_layer\\file_icon.gif'
        self.root = root
        self.init_main()
```

```
        self.root.protocol("WM_DELETE_WINDOW", self.exit_system)

    def init_main(self):
        self.menu_window()
        self.root.title("Searching app failures")
        self.root.geometry("650x400+300+100")
        self.root.resizable(False, False)

        welcome_label = Label(text="Welcome to system", font="Arial 32")

        choose_frame = LabelFrame(text="Choose file", height=200, width=200,
font="Arial 12")
        btn = tkinter.Button(choose_frame, command=self.open_dialog, image=self.icon)
        btn.pack()
        welcome_label.pack()
        choose_frame.pack()

    def open_dialog(self):
        file = filedialog.askopenfilename(
            initialdir='/',
            initialfile='tmp',
            filetypes=[("CSV", "*.csv")])
        if file != "":
            self.root.withdraw()
            cfw.FunctionWindow(self.root, file)
            dw.DataWindow(self.root, file)

    def menu_window(self):
        menu = Menu(self.root)
        self.root.config(menu=menu)
        file_menu = Menu(menu, tearoff=0)
        file_menu.add_command(label="Exit", command=self.exit_system)
        menu.add_cascade(label="File", menu=file_menu)

    def exit_system(self):
        if messagebox.askyesno("Exit", "Do you want to quit the application?"):
            self.quit()
```

Код для тестування системи

data_analyze_module_test.py

```
import unittest
import troubleshooting_system.data_analyze_layer.analyze_module as ad
import troubleshooting_system.data_analyze_layer.data_processing_module as dpm
import pandas as pd

class TestDataAnalyzeCase(unittest.TestCase):
    path = "E:\\Project\\Automated-troubleshooting-
system\\troubleshooting_system\\data_layer\\test_data.csv"

    def test_on_correct_return_data(self):
        self.assertIsNotNone(dpm.read_file(self.path), "Input data_layer is None")
        self.assertTrue(type(dpm.read_file(self.path)) == type(pd.DataFrame()))
        self.assertTrue(type(ad.find_statistics_param(self.path)),
type(pd.DataFrame()))
        lst_return = ad.data_error('Temperature', 'Failure',
dpm.read_file(self.path))
        self.assertTrue(len(lst_return) == 3)
        self.assertEqual(type(lst_return[0]), int)
        self.assertEqual(type(lst_return[1]), list)
        self.assertEqual(type(lst_return[2]), list)
        self.assertTrue(len(ad.get_colors(7)) == 7)
        self.assertEqual(type(ad.get_colors(7)), list)
        output_data = ad.dictionary_sort({"param1": 1, "param2": 2})
        self.assertTrue(len(output_data) == 2)

if __name__ == '__main__':
    unittest.main()
```

data_prediction_module_test.py

```
import math
import os
import unittest
from pathlib import Path
```

```
import troubleshooting_system.data_analyze_layer.prediction_module as pd
import troubleshooting_system.data_analyze_layer.data_processing_module as dpm
import pandas as pnd
class TestPredictionFunctionCase(unittest.TestCase):
    path = "E:\\Project\\Automated-troubleshooting-
system\\troubleshooting_system\\data_layer\\test_data.csv"



def test_read_file_on_correct_return(self):
        self.assertIsNotNone(dpm.read_file_prediction(self.path), "Input data_layer
is None")
        self.assertTrue(type(dpm.read_file_prediction(self.path)) ==
type(pnd.DataFrame()))

    def test_read_file_on_fill_empty(self):
        count_input_data =

        count_output_data = 0
        for el in pnd.read_csv(self.path)['Hours Since Previous Failure']:
nan = float(el)

if math.isnan(nan):
                count_input_data += 1
        for el in dpm.read_file_prediction(self.path)['Hours Since Previous
Failure']:
            nan = float(el)
            if math.isnan(nan):
                count_output_data += 1
        self.assertNotEqual(count_input_data, count_output_data)

    def test_correct_fill_median(self):
        current_val = pnd.read_csv(self.path)['Hours Since Previous
Failure'].median()
        expected_val = dpm.read_file_prediction(self.path)['Hours Since Previous
Failure'][1]
        self.assertEqual(current_val, expected_val)

    def test_replace_value_data_correct_result(self):
        result_array = pd.replace_value_data(pnd.read_csv(self.path, index_col=0),
['Temperature', 'Humidity'] "Failure")
```

```
        self.assertEqual(len(result_array), 2)

    def test_replace_value_data_incorrect_input_data(self):
        check_none = pd.replace_value_data(pnd.read_csv(self.path), ['Temperature',
'Error'], "Failure")
        self.assertIsNone(check_none)
        check_none = pd.replace_value_data(pnd.read_csv(self.path), ['Temperature'],
"Incorrect value")
        self.assertIsNone(check_none)

    def test_prediction_return_type(self):
        check_none = pd.prediction(pnd.read_csv(self.path), ['Temperature', 'Error'],
"Failure", 1)
        self.assertIsNone(check_none)
        check_none = pd.prediction(pnd.read_csv(self.path), ['Temperature'],
"Failure", 2)
        self.assertIsNotNone(check_none)
        self.assertEqual(type(check_none), str)

    def test_out_predict_data_save_file(self):
        home = str(Path.home())
        pd.prediction(pnd.read_csv(self.path), ['Temperature'], "Failure", 1)
        pd.prediction(pnd.read_csv(self.path), ['Temperature'], "Failure", 2)
        self.assertTrue(os.path.exists(home + "\\prediction_data_lr.csv"))
        self.assertTrue(os.path.exists(home + "\\prediction_data_rf.csv"))

    def test_save_model_file(self):
        home = str(Path.home())
        count_file_after = 0
        list_dir = os.listdir(home)
        for file in list_dir:
            if file.endswith('.pkl'):
                os.remove(home + "\\" + file)
        pd.prediction(pnd.read_csv(self.path), ['Temperature'], "Failure", 1)
        for file in list_dir:
            if file.endswith('.pkl'):
                count_file_after += 1
        self.assertTrue(count_file_after > 0)
if __name__ == '__main__':
    unittest.main()
```